SQL Server : Scale out & performance

Christophe Laporte

#SQLSatMadrid

PASS SQL saturday

#568 | MADRID 2016

Apologies – French speaker

Christophe Laporte

@conseilit

Microsoft SQL Server

~ since 1997
6.5 <= SQL Server <= 2016

Microsoft CERTIFIED Master

christophe_laporte@hotmail.fr

http://conseilit.wordpress.com/

MVP Microsoft® Most Valuable Professional

PASS SQL saturday
#568 | MADRID 2016

BIG Thanks to SQLSatMadrid Sponsors

Hewlett Packard Enterprise

#SQLSatMadrid

4 Sponsor Sessions at **11:40**

Don’t miss them, they might be getting distributing some **awesome prizes**!

- HPE
- SolidQ
- KABEL
- TSD Consulting

Also **BIG Raffle prizes** at the end of the event provided by:

Plainconcepts, SolidQ, Kabel, TSD Consulting, Pyramid Analytics & sqlpass.es

Agenda

- Scaling for performance

What is scaling { up | out }

- *Scalability* is the ability of an application to efficiently use more resources in order to do more useful work.
- Scale Up
  - Bigger box

- Scale Out
  - Expending to multiple servers

- Main idea
  - Run queries faster
  - Run more queries at the same time

Scale up

- Historically
  - Processors speed
  - More cores
  - More RAM
  - Disks faster and faster (SSD/PCI/NVMe and NVDIMM)
- Problem with scaling up
  - Fault tolerance very expensive
  - Scaling on-demand
    - Support high-end workload
    - Workload changes

But …

- Locks & Latches
  - RCSI
  - In memory tables
- Only 1 TempDB
  - Adding more files to decrease contention (GAM / SGAM / PFS)
  - SQL 2014 : TempDB eager write
  - SQL 2016 : new Query hint NO_PERFORMANCE_SPOOL
- 1 transaction log file / Limit of Log Manager
  - Delayed durability
  - Schema only durability for Hekaton tables

Scale out

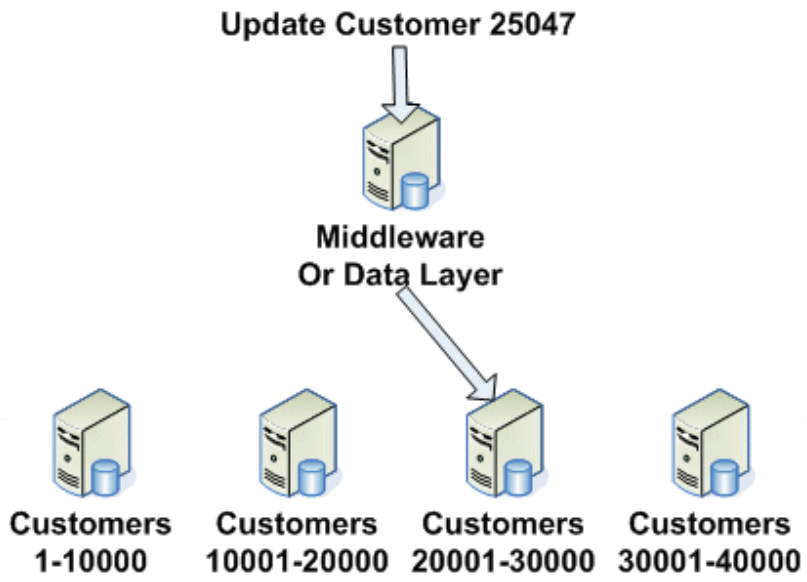
- Scale out - the future ?
  - SQL
  - NoSQL
- Scale out often means fault tolerant
- Small DBs / instances
  - Easier to manage
  - Lower cost / commodity hardware
  - Geographically dispersed
    - Data nearest of the end-user
    - Network latency

How to scale out ? (the real agenda !)

- Middle Tier
- Distributed partitioned view
- Replication
- Log shipping
- Scalable Shared Databases
- Partitioned tables
- Availability groups
- Service broker

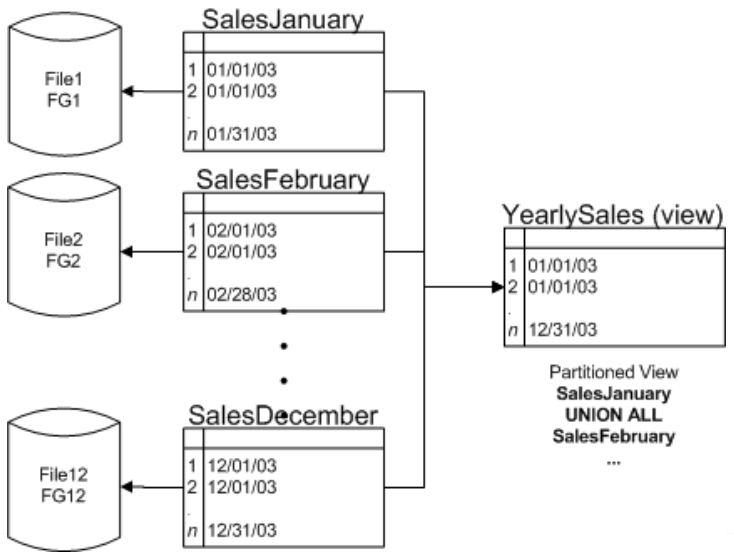
Middle Tier/ data layer

- Data-dependent routing from middleware
- Table / database / instance chosen by code
- Boost performance with caching
- Can be combined



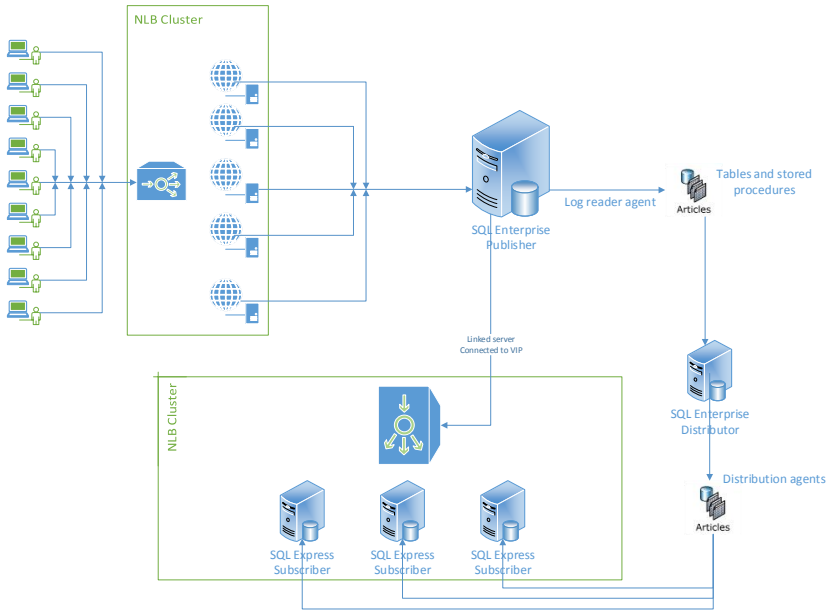
(Distributed) Partitioned Views

- Can be spread across multiple servers
- Local joins to not move data across servers
- Smaller tables : easy to manage (Index, stats, checkdb)
- Horizontal partitioning
- Take care to restore point
- Old fashion (SQL 2000)
- Demo 1



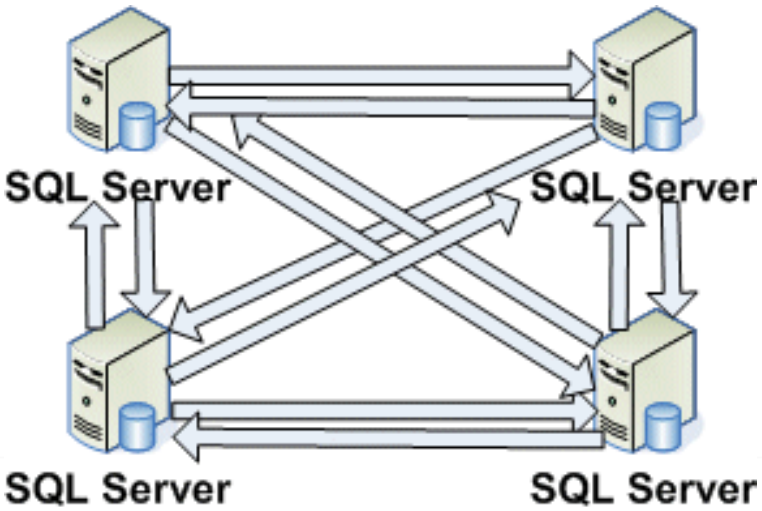
Replication

- Read-only
  - Easy to setup and manage
  - 1 publisher / N subscribers
  - N publishers / 1 subscriber
  - transactional replication
- Read-Write
  - Merge replication
  - 1..N publishers / 1..N subscribers
  - Slightly more complex to setup and manage
- Require additional feature to load balance

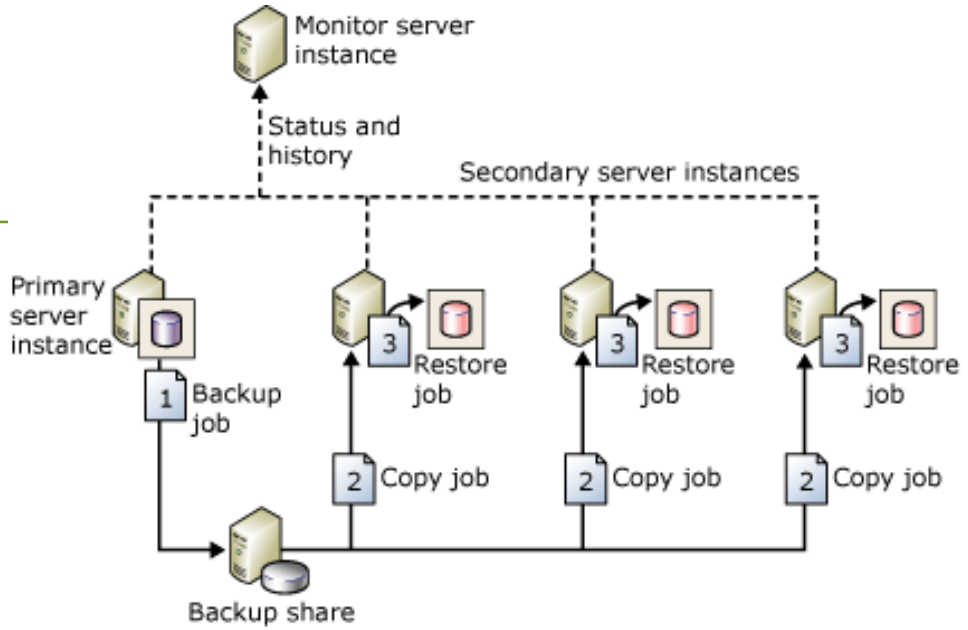


Peer-To-Peer Replication

- Enterprise edition
- May be geographically dispersed
- Gaps / identity columns
- Update frequency moderate



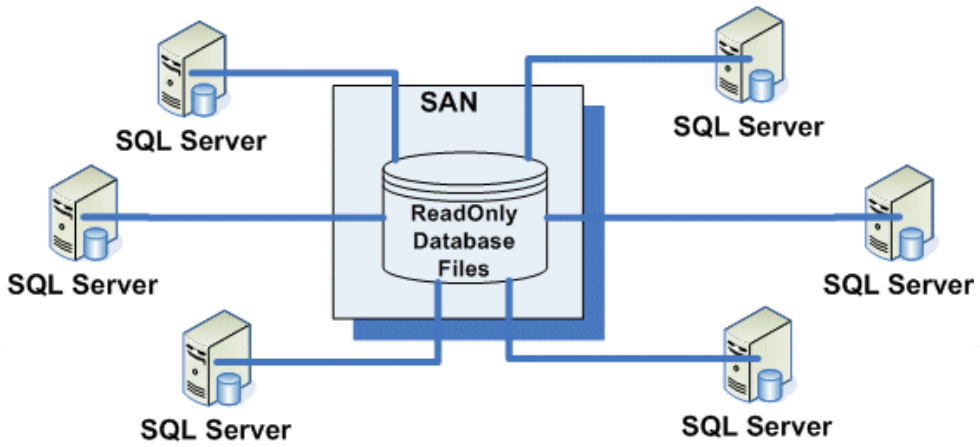
Log Shipping



- Read-only
- all editions
- Large # of secondary
- Latency to get up-to-date data
- May be geographically dispersed
- Users disconnected during restores
- Require additional feature to load balance

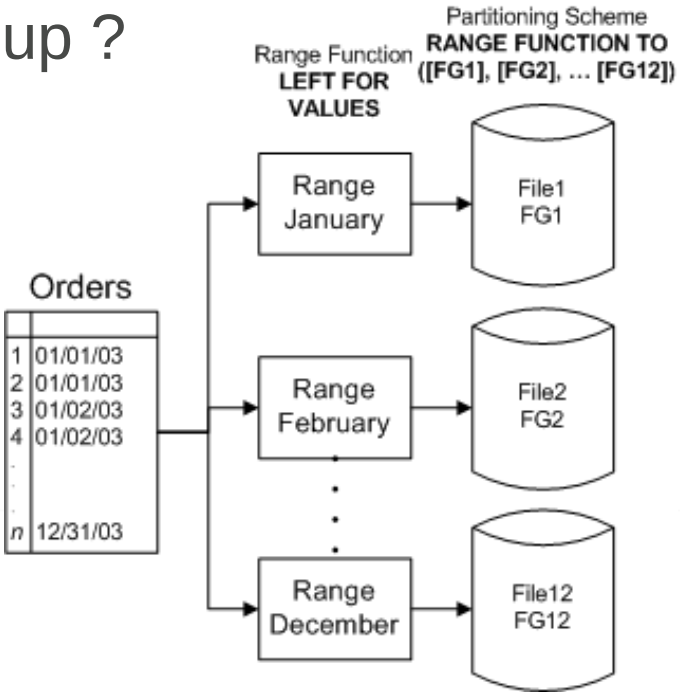
Scalable Shared Databases

- Enterprise edition
- Read-only
  - Updating the DB may be difficult / decrease availability
- Require additional feature to load balance



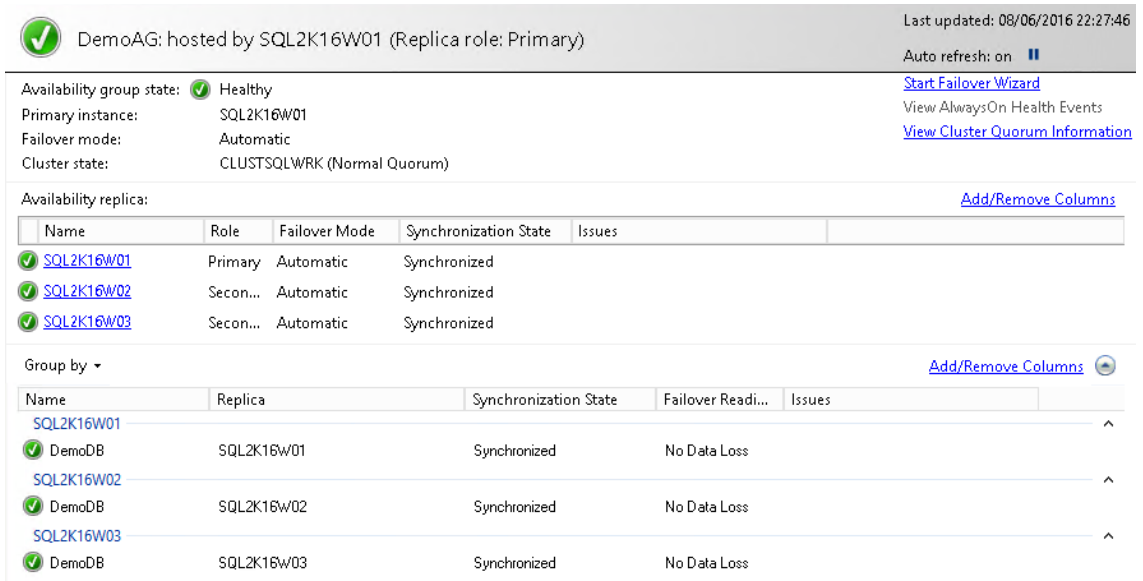
Partitioned Tables

- Lock reduction (lock escalation at partition level)
- Deal with different kind of storage / compression level
- Truncate partition (new in SQL 2016)
- Resolves page latch contention issues
- Scaling out or scaling up ?
- Edition enterprise
- Demo 2



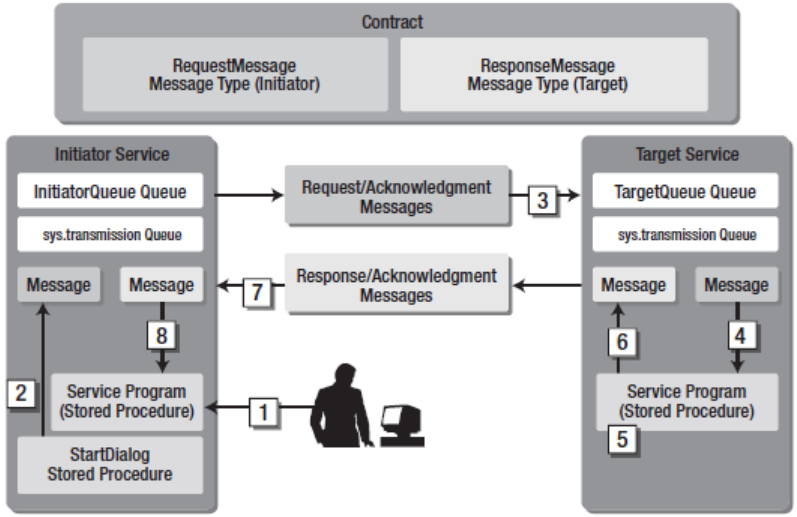
Availability Groups – RO load Balancing

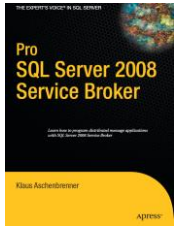
- Read Only Load Balancing : New in SQL Server 2016
- Easy setup & does not require high skills on system administration
- Domain agnostic
- Up to 8 RO replicas
- 3 synchronous
- Enterprise edition
- Demo 3



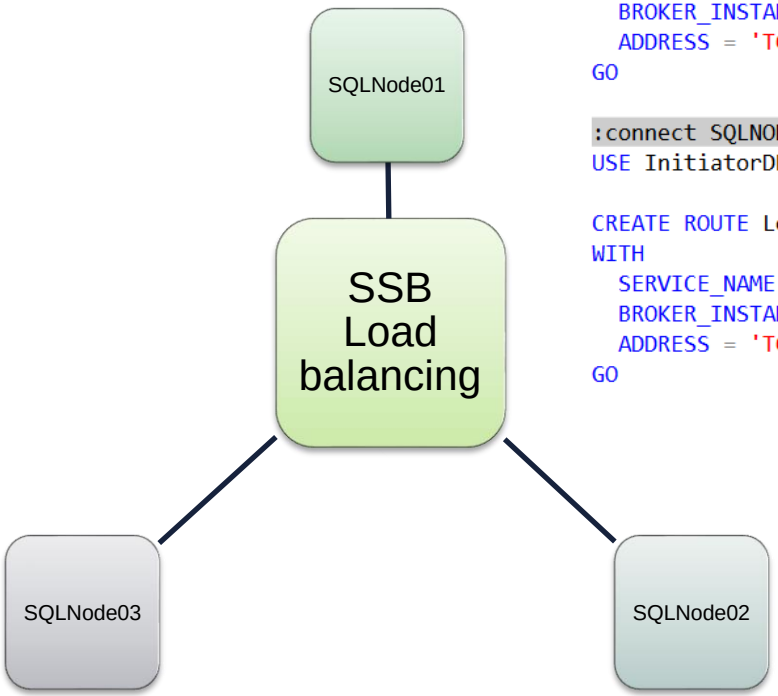
Service Broker

- Asynchronous operations
- Sens message to a target
  - Same database
  - Same instance different database
  - Different instance
- Routing between instances
  - Load balancing
  - Multicasting





Demo 4 – Service broker & NLB

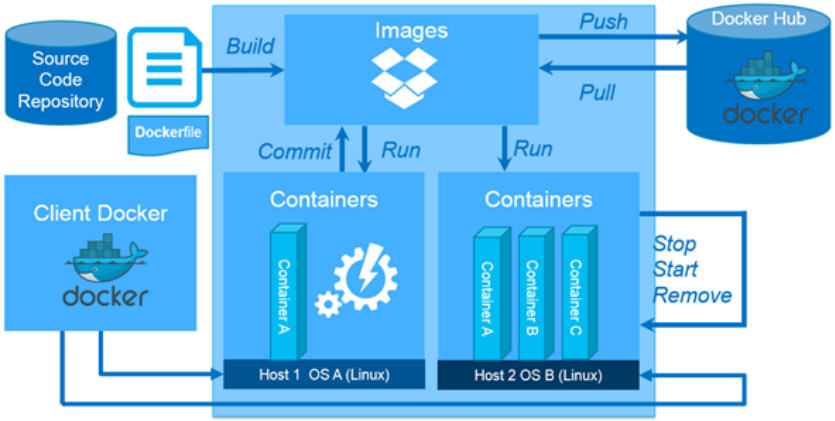


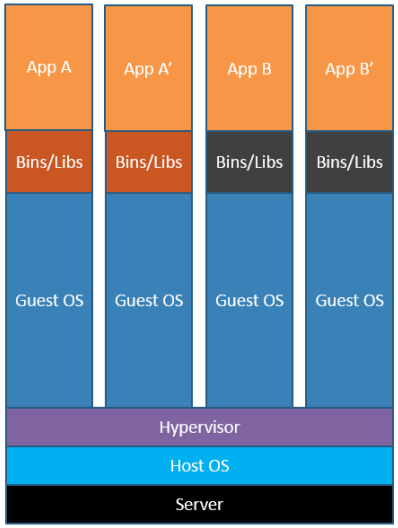

```
:connect SQLNODE01
USE InitiatorDB

CREATE ROUTE LoadBalancingRouteSQLNode02
WITH
  SERVICE_NAME = 'TargetService',
  BROKER_INSTANCE = '6E7B09CF-897E-4595-9031-9959FEB9B325',
  ADDRESS = 'TCP://SQLNODE02:7022'
GO

:connect SQLNODE01
USE InitiatorDB

CREATE ROUTE LoadBalancingRouteSQLNode03
WITH
  SERVICE_NAME = 'TargetService',
  BROKER_INSTANCE = 'A28094A4-B464-4198-8718-CFCCA8CCFA23',
  ADDRESS = 'TCP://SQLNODE03:7022'
GO
```

Same service name
= Load Balancing

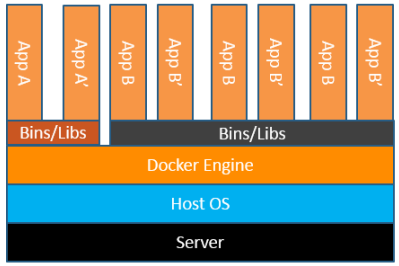
Windows Server containers / Docker

- Use cases : Dev / Test / Prod
- Lightweight SQL Server instance inside container
- Quick response to load increase
- Scaling out if combined with other solutions
  - Third party load balancer
  - (/!\ Replication /!\ )
  - Service Broker





Virtual machines versus containers



IT Demo – SQL Server Express in a container

Run container

Detached mode

Volumes redirection Host:guest

```
docker run --name sqlsaturday -d -p 568:1433 -v c:\mssql\sqlsaturday:c:\sql sqlexpress
```

Container name

Ports redirection Host:Guest

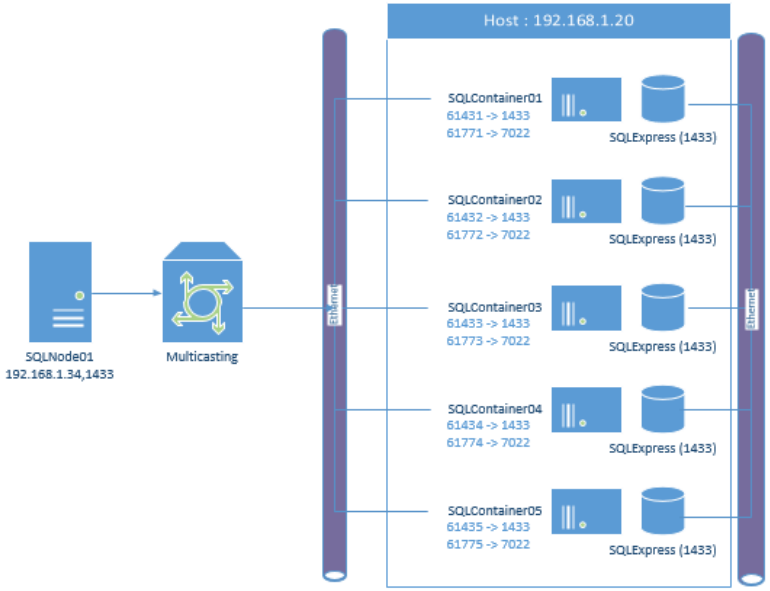
Image name

```
PS C:\Users\Administrator> docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS              PORTS                                                   NAMES
aea5623fd5cd        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61439->1433/tcp, 0.0.0.0:61779->7022/tcp        sql09
dd76aa08926c        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61438->1433/tcp, 0.0.0.0:61778->7022/tcp        sql08
bab00c25e867        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61437->1433/tcp, 0.0.0.0:61777->7022/tcp        sql07
3a3299f4e24d        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61435->1433/tcp, 0.0.0.0:61775->7022/tcp        sql05
860b0f630541        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61434->1433/tcp, 0.0.0.0:61774->7022/tcp        sql04
e0d46a11f1ff        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61433->1433/tcp, 0.0.0.0:61773->7022/tcp        sql03
031e615766a7        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61432->1433/tcp, 0.0.0.0:61772->7022/tcp        sql02
bc0da018328e        sqlexpress          "cmd /S /C 'powershel"   3 days ago          Up 3 days           0.0.0.0:61431->1433/tcp, 0.0.0.0:61771->7022/tcp        sql01
```

Demo 5 – Service Broker & Multicasting with Docker

```
BEGIN TRANSACTION;

    BEGIN DIALOG @DialogHandleSQL01
     FROM SERVICE InitiatorService
     TO SERVICE   N'TargetServiceSQL01'
     ON CONTRACT  SampleContract
     WITH ENCRYPTION = OFF;

    BEGIN DIALOG @DialogHandleSQL02
     FROM SERVICE InitiatorService
     TO SERVICE   N'TargetServiceSQL02'
     ON CONTRACT  SampleContract
     WITH ENCRYPTION = OFF;

    BEGIN DIALOG @DialogHandleSQL03
     FROM SERVICE InitiatorService
     TO SERVICE   N'TargetServiceSQL03'
     ON CONTRACT  SampleContract
     WITH ENCRYPTION = OFF;

    BEGIN DIALOG @DialogHandleSQL04
     FROM SERVICE InitiatorService
     TO SERVICE   N'TargetServiceSQL04'
     ON CONTRACT  SampleContract
     WITH ENCRYPTION = OFF;

    BEGIN DIALOG @DialogHandleSQL05
     FROM SERVICE InitiatorService
     TO SERVICE   N'TargetServiceSQL05'
     ON CONTRACT  SampleContract
     WITH ENCRYPTION = OFF;


    SEND ON CONVERSATION (
        @DialogHandleSQL01,@DialogHandleSQL02,@DialogHandleSQL03,@DialogHandleSQL04,@DialogHandleSQL05
    )
         MESSAGE TYPE RequestMessage (@RequestMsg);

    SELECT @RequestMsg AS SentRequestMsg;

COMMIT TRANSACTION;
```

Multiple Begin Dialog
Single Send on Conversation



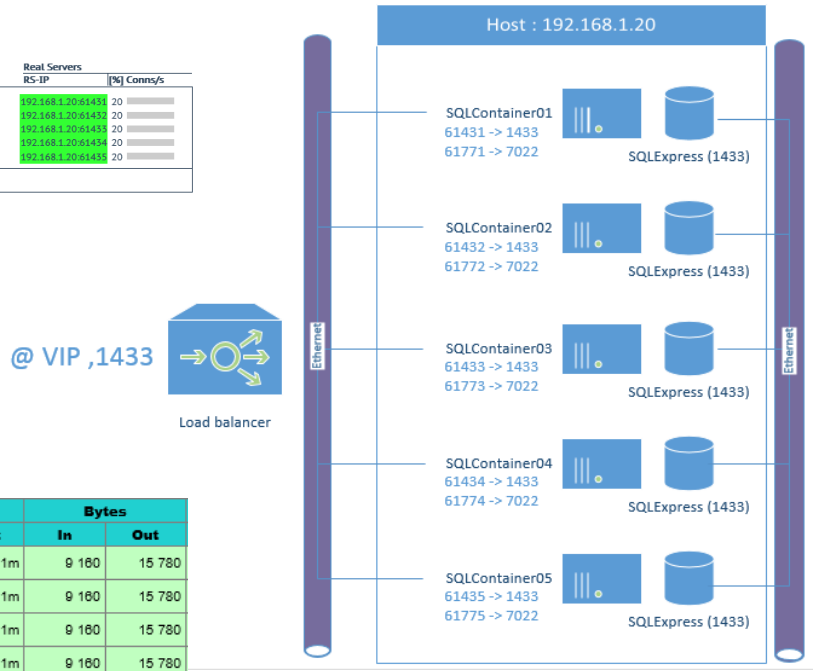
#SQLSatMadrid

How to load balance workload

- Windows NLB
- Windows 2016 Software Load Balancing (SLB) for SDN
- 3rd party hardware
  - Kemp
  - F5
  - Cisco
  - Citrix
  - Radware
- 3rd party software
  - Kemp
  - Pfsense
  - Scalearc
  - Nginx
  - …

Demo 6 – load balancing with containers

Name	Virtual IP Address	Protocol	Status	Total Conns	Last 60 Sec	5 Mins	30 Mins	1 Hour	Active Conns	Current Rate Conns/s	Real Servers RS-IP	[%] Conns/s
1 NLBMSSQL	192.168.1.128:1433	tcp	Up	557	100	150	150	150	50	10	192.168.1.20:61431 192.168.1.20:61432 192.168.1.20:61433 192.168.1.20:61434 192.168.1.20:61435	20 20 20 20 20
1	System Total Conns			557	100	150	150	150	50	10 /sec		



		Queue			Session rate			Sessions						Bytes	
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	LbTot	Last	In	Out
☐	Conteneur01	0	0	-	0	10		0	10	-	10	10	4h21m	9 160	15 780
☐	Conteneur02	0	0	-	0	10		0	10	-	10	10	4h21m	9 160	15 780
☐	Conteneur03	0	0	-	0	10		0	10	-	10	10	4h21m	9 160	15 780
☐	Conteneur04	0	0	-	0	10		0	10	-	10	10	4h21m	9 160	15 780
☐	Conteneur05	0	0	-	0	10		0	10	-	10	10	4h21m	9 160	15 780
	Backend	0	0		0	50		0	50	200	50	50	4h21m	45 800	78 900

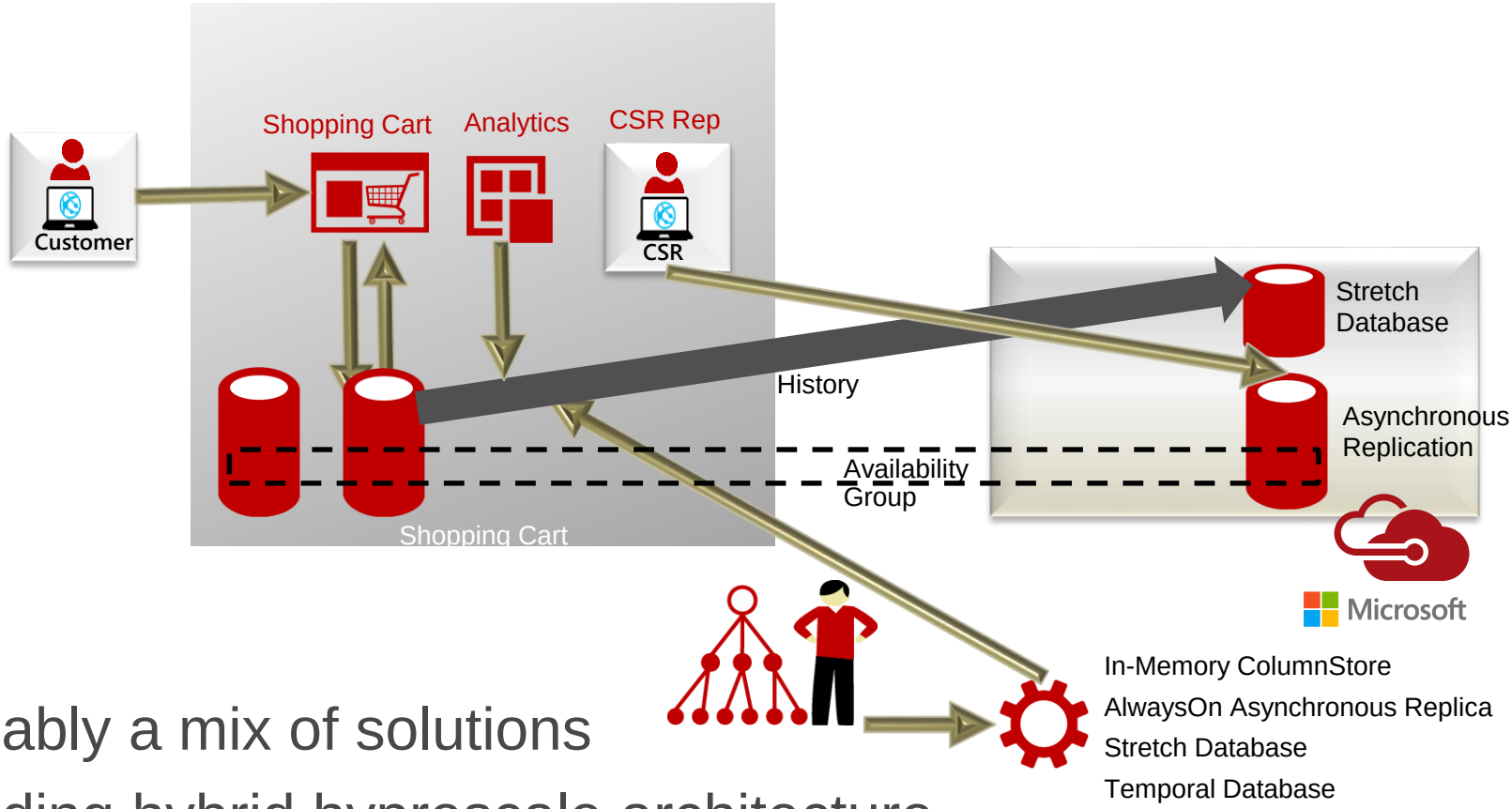
SQL Server on Azure …

- Not covered in this session
- By design
  - SQL DataWarehouse
    - Elastic data warehouse as a service
  - Azure SQL Databases
    - Sharding / Federation
    - Elastic pools
- Scale up/down
  - Change service-tiers for a given database
- Scale out
  - Add Remove databases

Scaling : remember some basics

- A clustered index on an integer column does not scale !
  - Unless partitioning
- A GUID / uniqueidentifier does
  - But introduce fragmentation

The future ?



- Probably a mix of solutions
- Including hybrid hyprescale architecture

Q/A

- Hope you enjoyed
- Thanks for attending
- Q/A