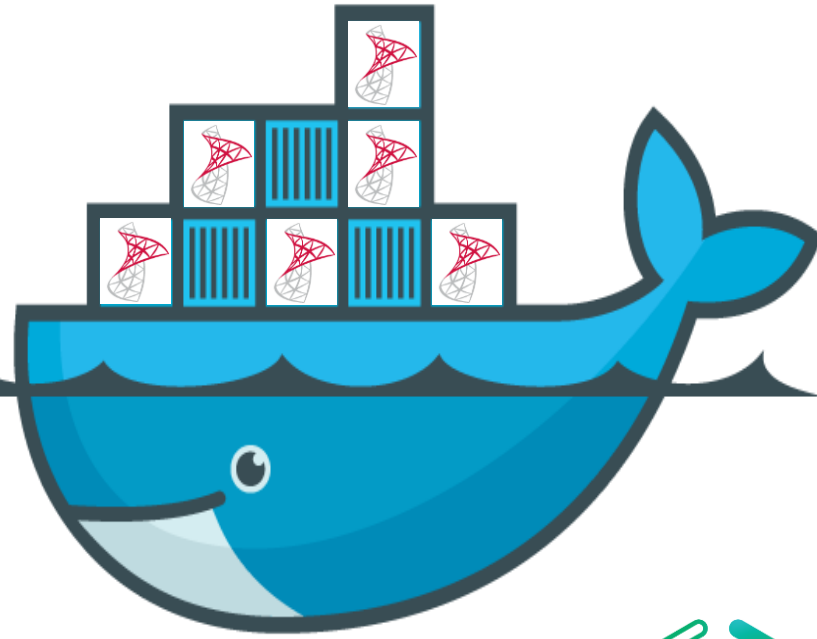
SQL Server & Docker
Les premiers pas



Christophe LAPORTE – SQL Server MVP/MCM

SQL Saturday 762 – Paris 2018

Thank you Sponsors

DBPLUS
better performance

Data by Design

CONCATSKILLS

criteo

Conseil IT

Microsoft

You are Community

DATACROSSROAD

AZEO
talents & technology

PASS

Christophe LAPORTE

@conseilit

Microsoft SQL Server ~ since 1997 : SQL 6.5 / WinNT4

christophe_laporte@hotmail.fr

http://conseilit.wordpress.com/

Microsoft CERTIFIED Master

Microsoft CERTIFIED Trainer

MVP Microsoft Most Valuable Professional

- Audit
- Conseil
  - Infrastructure / Architecture
  - Virtualisation / Cloud
  - Haute disponibilité
  - Performance / Optimisation
  - Dépannage / Migrations
- Formations
- Remote DBA
- Hébergement BDD

Thank you Sponsors

DBPLUS
better performance

Data by Design

CONCATSKILLS

criteo

Conseil IT

Microsoft

You are Community

DATACROSSROAD

AZEO
talents & technology

PASS

Agenda

- Micro services et containers
- Installing Docker
- Creating my first container
- Running a SQL Server container
- Customizing a container
- Creating multi-container Applications
- Q/A

Discovery session

Docker ?

Platform

For developers and sysadmins

Build images

Image : Read-only template with instructions for creating a container

Run containers

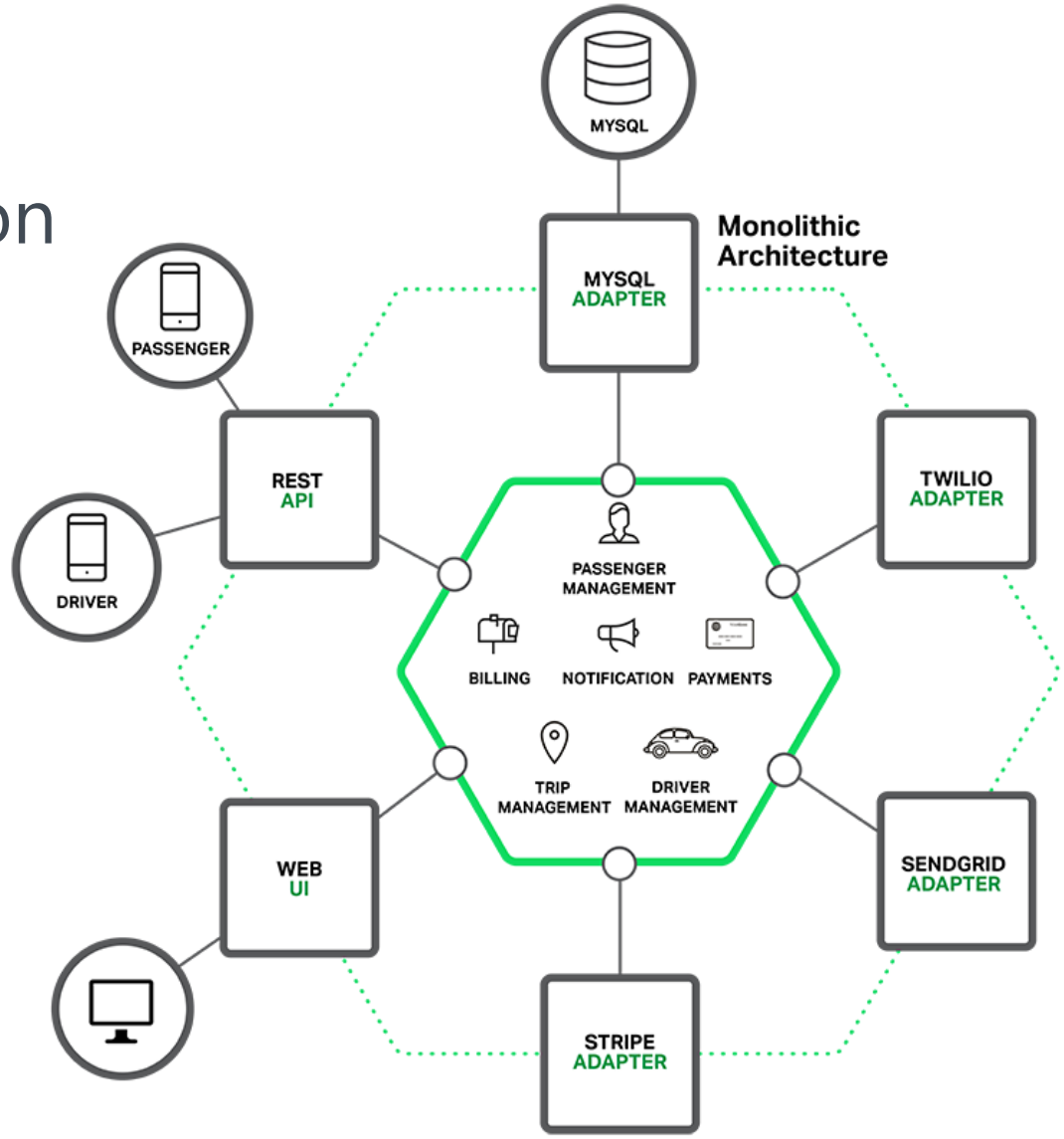
Container : Runnable instance of an image, isolated from surroundings

Next step for virtualization

Why containers ?

The end of monolithic application

- Hard to maintain
- Delay between releases
- Complex deployment



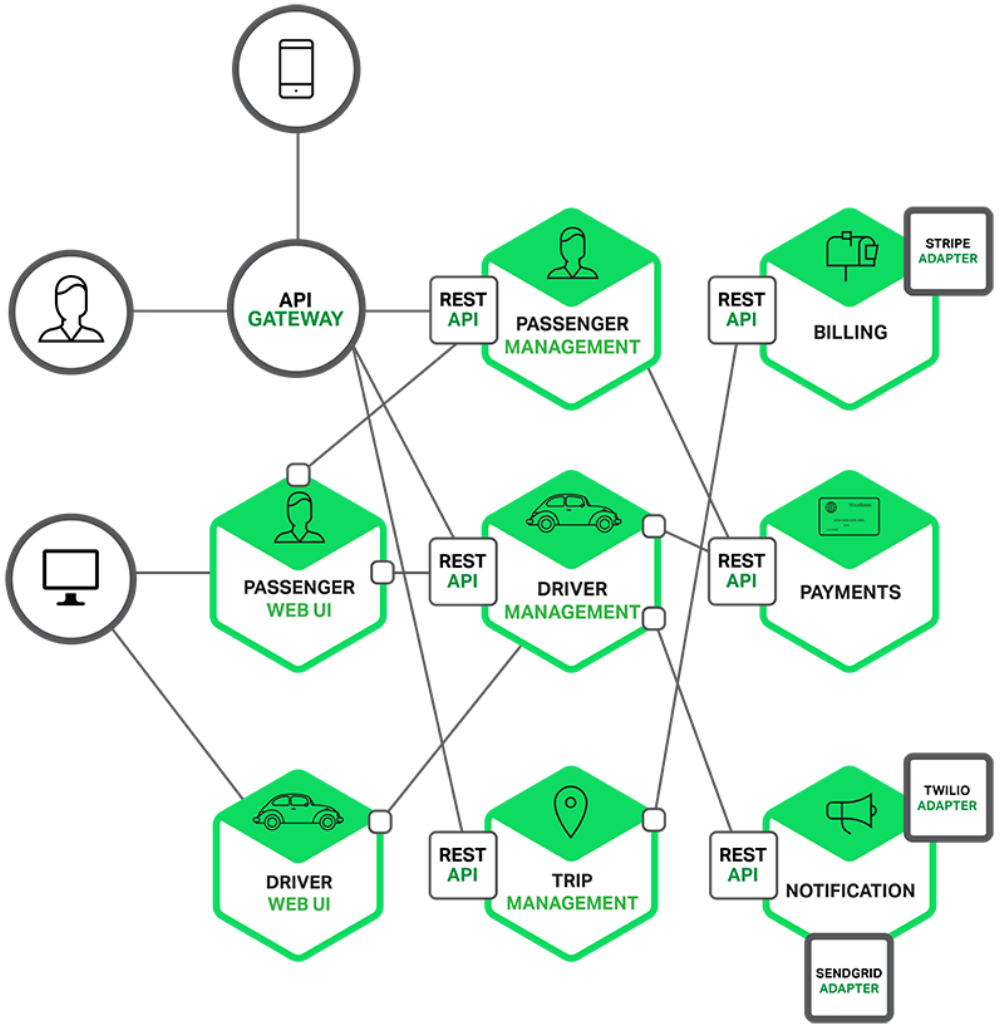
Micro services

New way to develop applications

- Lightweight pieces of SW evolving independently
- 1, 10s or 100s of containers composed as a single application

DevOps

- Continuous integration and deployment made easier
- Accelerate development and deployment



Demo – Installing Docker

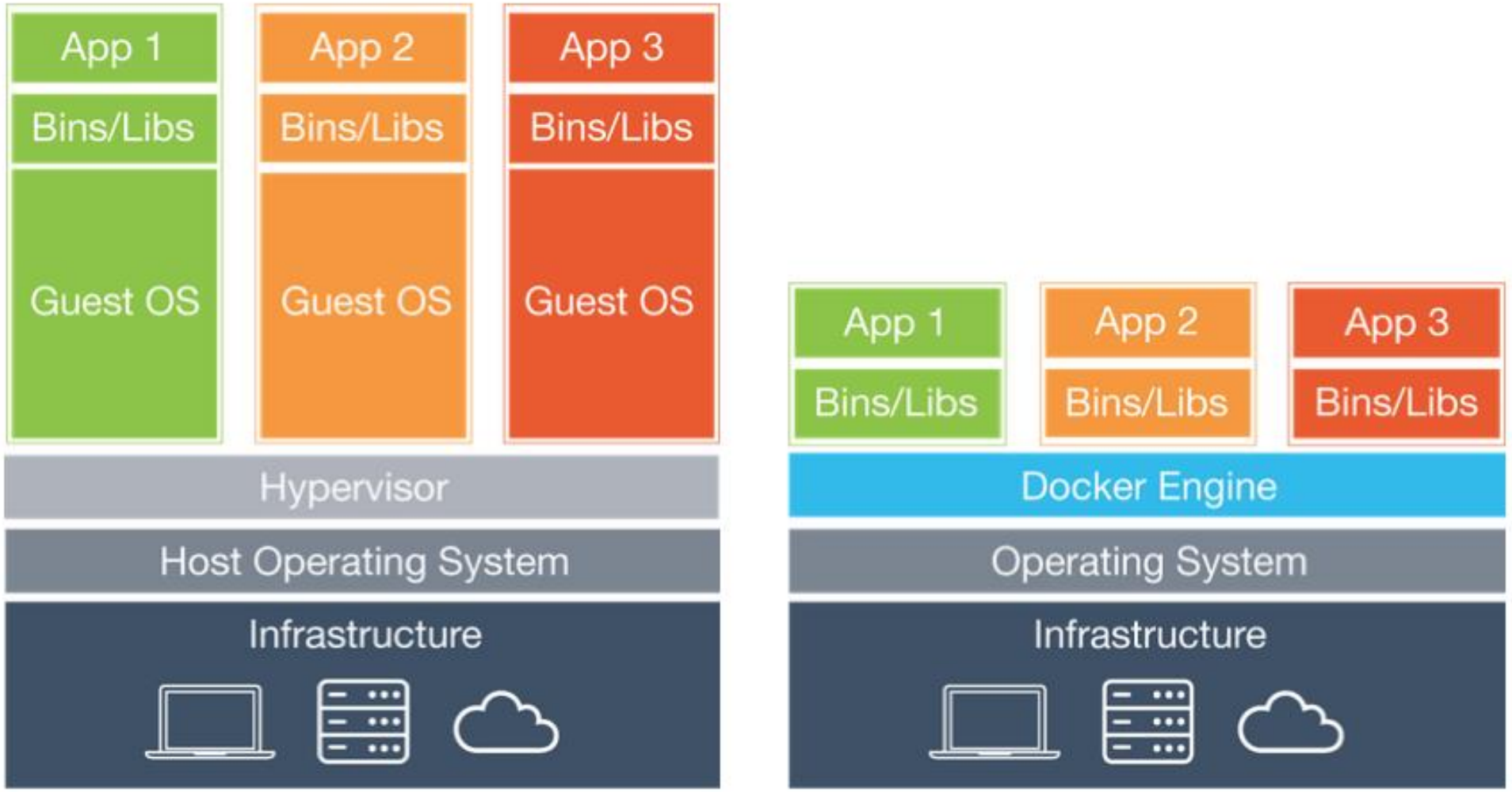

```
Install-Module -Name DockerMsftProvider -Repository PSGallery -Force
Install-Module -Name DockerMsftProvider -Force
Install-Package -Name docker -ProviderName DockerMsftProvider -Force
Restart-Computer -Force
```


```
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
```

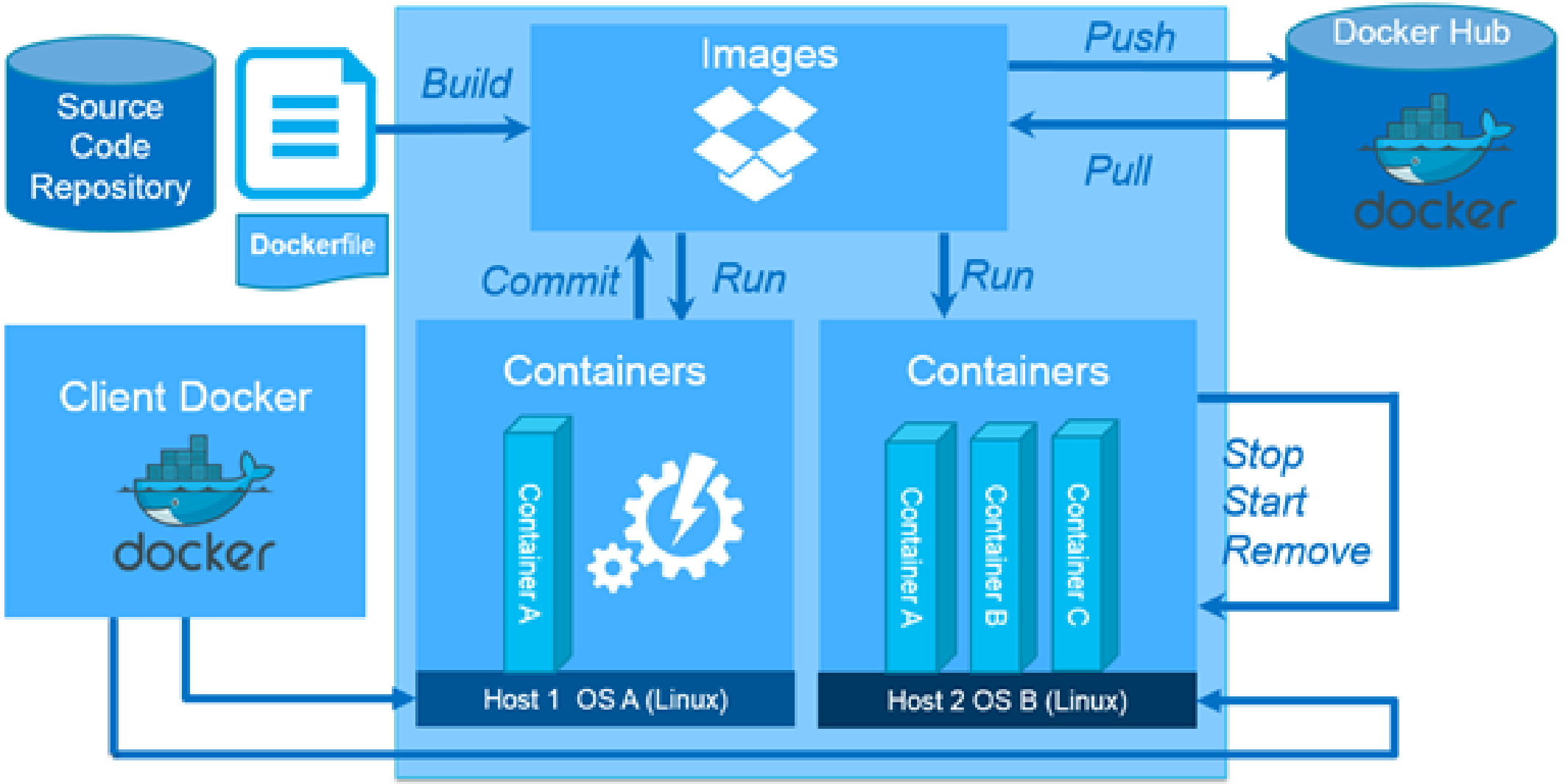
Windows Server 2016 Datacenter - with Containers

Docker on Ubuntu Server

Virtualization vs containerization



Docker architecture



Demo – My first Docker commands

```
## List Docker CLI 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

## List Docker networks
docker network ls
docker network inspect bridge
```

Docker commands vs traditional software

"traditional" software	Docker command
Find the binaries for setup …	Docker search
Download and mount ISO file	Docker pull
Create ISO / Zip file	Docker build
Install the software	Docker create
Run the software	Docker start
Download, install and run software (all-in-one)	Docker run
Stop the software	Docker stop
Uninstall the software	Docker rm

Demo – my first container

```
# pull a Windows Server Core image
docker pull microsoft/windowsservercore

# download SQL Server express binaries
Invoke-WebRequest -Uri "https://go.microsoft.com/fwlink/?linkid=829176" -OutFile sqlexpress.exe

# create a container
docker run --name MyFirstContainer -it microsoft/windowsservercore powershell

# copy somes files into the container
docker cp sqlexpress.exe MyFirstContainer:/Install/sqlexpress.exe

# expand the binaries
/install/sqlexpress.exe /q /x:/install/setup

# install SQL Server
/install/setup/setup.exe /Q /ACTION=Install `
  /INSTANCENAME=MSSQLServer `
  /FEATURES=SQLEngine `
  /UPDATEENABLED=1 `
  /SECURITYMODE=SQL /SAPWD=Password1! `
  /SQLSVCACCOUNT="NT AUTHORITY\System" `
  /SQLSYSADMINACCOUNTS="BUILTIN\ADMINISTRATORS" `
  /IACCEPTSQLSERVERLICENSETERMS
```

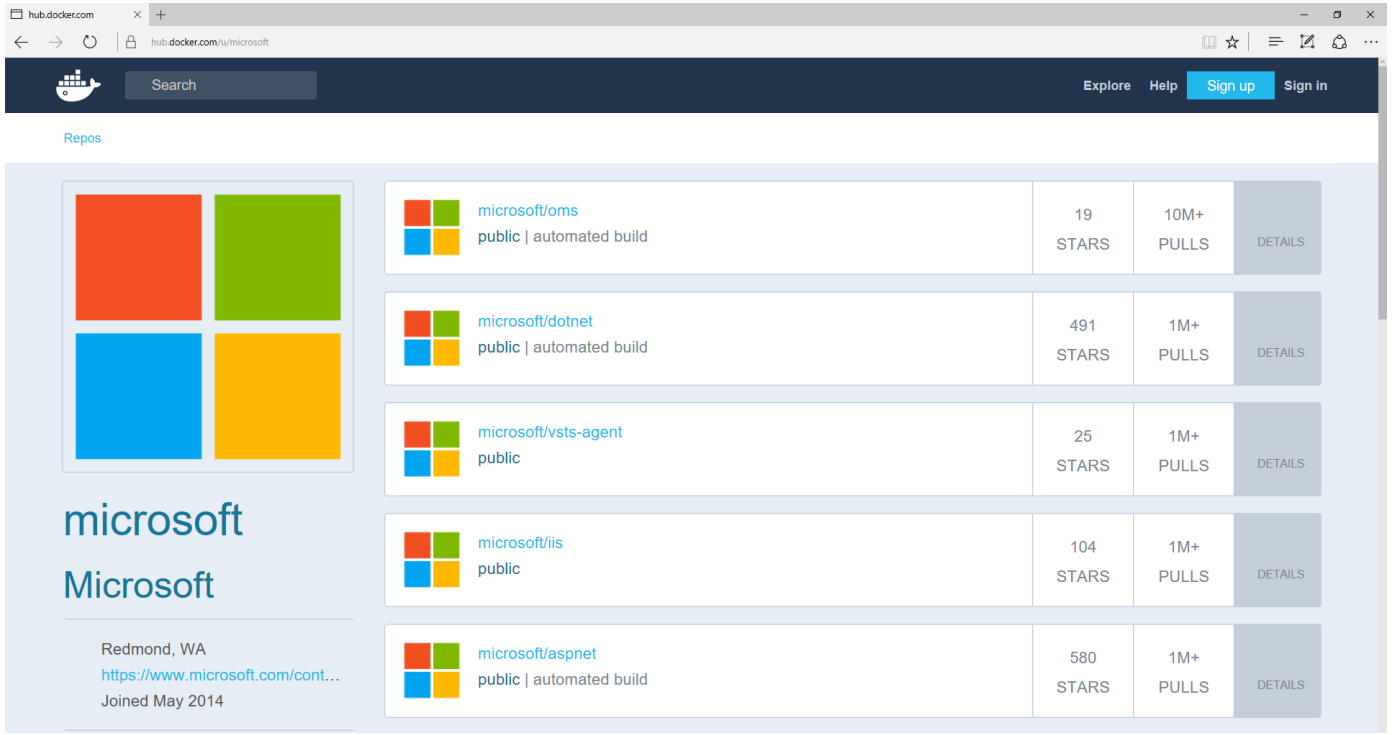
Docker Hub – the easiest way to get an image

Images repository

- Search images
- Pull images
- Push images

Microsoft official images

IIS, SQL Server, MongoDB, …



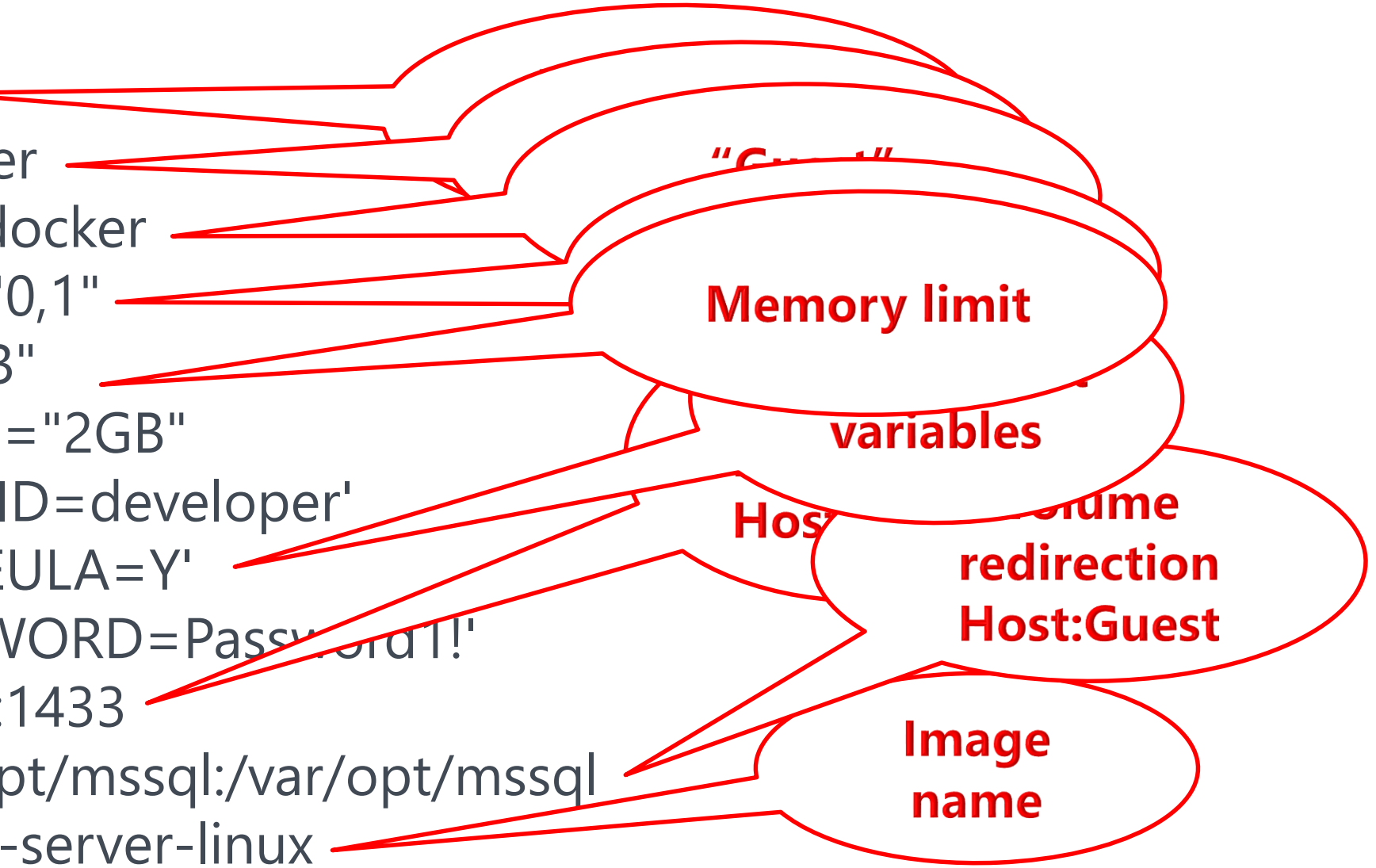
Demo – pulling a SQL Server image


```
# Pull SQL Server Express image from Docker Hub
# https://hub.docker.com/r/microsoft/mssql-server-windows-express/
docker pull microsoft/mssql-server-windows-express

# Pull SQL Server Developer image from Docker Hub
# https://hub.docker.com/r/microsoft/mssql-server-windows-developer/
docker pull microsoft/mssql-server-windows-developer
```


```
# Pull SQL Server "all editions" image from Docker Hub
# https://hub.docker.com/r/microsoft/mssql-server-linux/
sudo docker pull microsoft/mssql-server-linux
```

Docker run command line

```
docker run --detach
        --name sqldocker
        --hostname sqldocker
        --cpuset-cpus="0,1"
        --memory="2GB"
        --memory-swap="2GB"
        --env 'MSSQL_PID=developer'
        --env 'ACCEPT_EULA=Y'
        --env 'SA_PASSWORD=Passw0rd1!'
        --publish 14331:1433
        --volume /var/opt/mssql:/var/opt/mssql
        microsoft/mssql-server-linux
```



Demo – Running a container


```
# 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
```

Healthcheck

- Docker stat
- Container heath check
  - Dockerfile instruction

    HEALTHCHECK CMD ["sqlcmd", "-Q", "select 1"]
  - Docker compose

```
healthcheck:
  test: ["CMD", "/opt/mssql-tools/bin/sqlcmd", "-Usa", "-PPassword1!", "-Q", "select 1"]
```
  - Runtime

    DOCKER RUN --health-cmd 'sqlcmd -Q "select 1 " ’
  - Inspect the status

    DOCKER PS

    Docker inspect -f '{{json .State.Health.Status}}’ containername

Demo Healthcheck

Dockerfile

```
FROM microsoft/mssql-server-linux
HEALTHCHECK --interval=5s CMD ["/opt/mssql-tools/bin/sqlcmd", "-Usa", "-PPassword1!", "-Q", "select 1"]
```

Runtime

```
docker run  --detach \
            --name sqldocker \
            --hostname sqldocker \
            --env 'MSSQL_PID=developer' \
            --env 'ACCEPT_EULA=Y' \
            --env 'SA_PASSWORD=Password1!' \
            --health-cmd '/opt/mssql-tools/bin/sqlcmd -Usa -PPassword1! -Q "select 1"' \
            --health-interval '5s' \
            --publish 1433:1433 microsoft/mssql-server-linux
```


Monitoring

DIY

collectd / influxdb / grafana

https://github.com/Microsoft/mssql-monitoring

3rd party

Datadog

New Relic

Dynatrace

dockerfile file

Build customized image

Script based

- # Comments
- FROM (1..N)
- LABEL MAINTAINER
- ENV (1..N)
- COPY / ADD (1..N)
- RUN (1..N)
- CMD (1)

```
FROM microsoft/windowsservercore

LABEL maintainer "Perry Skountrianos"

# Download Links:
ENV exe "https://go.microsoft.com/fwlink/?linkid=840945"
ENV box "https://go.microsoft.com/fwlink/?linkid=840944"

ENV sa_password="_" \
    attach_dbs="[]" \
    ACCEPT_EULA="_" \
    sa_password_path="C:\ProgramData\Docker\secrets\sa-password"

SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop'; $ProgressPreference = 'SilentlyContinue';"]

# make install files accessible
COPY start.ps1 /
WORKDIR /

RUN Invoke-WebRequest -Uri $env:box -OutFile SQL.box ; \
        Invoke-WebRequest -Uri $env:exe -OutFile SQL.exe ; \
        Start-Process -Wait -FilePath .\SQL.exe -ArgumentList /qs, /x:setup ; \
        .\setup\setup.exe /q /ACTION=Install /INSTANCENAME=MSSQLSERVER /FEATURES=SQLEngine /UPDATEENABLED=0 /SQLSVCACCOUNT='NT AUTHORITY\System'
        Remove-Item -Recurse -Force SQL.exe, SQL.box, setup

RUN stop-service MSSQLSERVER ; \
        set-itemproperty -path 'HKLM:\software\microsoft\microsoft sql server\mssql14.MSSQLSERVER\mssqlserver\supersocketnetlib\tcp\ipall' -name
        set-itemproperty -path 'HKLM:\software\microsoft\microsoft sql server\mssql14.MSSQLSERVER\mssqlserver\supersocketnetlib\tcp\ipall' -name
        set-itemproperty -path 'HKLM:\software\microsoft\microsoft sql server\mssql14.MSSQLSERVER\mssqlserver\' -name LoginMode -value 2 ;

HEALTHCHECK CMD [ "sqlcmd", "-Q", "select 1" ]

CMD .\start -sa_password $env:sa_password -ACCEPT_EULA $env:ACCEPT_EULA -attach_dbs \"$env:attach_dbs\" -Verbose
```

```
docker build -t MyCustomSQLServerImage .
```

Demo – Creating custom image

```
# pull the base image from Docker Hub if not find locally
FROM microsoft/mssql-server-linux

ENV SA_PASSWORD=saTemp0r@ryP@ssw0rd
ENV ACCEPT_EULA=Y

# copy some files and folders into the container/image
COPY entrypoint.sh entrypoint.sh
COPY sqlPostInstallStartup.sh sqlPostInstallStartup.sh
COPY ./sqlscripts ./sqlscripts

RUN chmod +x ./sqlPostInstallStartup.sh

# adding healthcjeck to the container at design time
HEALTHCHECK --interval=5s CMD ["/opt/mssql-tools/bin/sqlcmd", "-UDockerHealthCheck", "-PDockerHealthCheck", "-Q", "select 1"

# and finally run the entry point to start SQL Server
# and run the postinstall bash to run all T-SQL scripts
CMD /bin/bash ./entrypoint.sh $sa_password
```

Demo – Creating custom image

```
# start the postinstall script then start the sqlserver
./sqlPostInstallStartup.sh "$1" & /opt/mssql/bin/sqlservr
```

```
#wait for the SQL Server to come up
sleep 20s

#run the setup script to create the DB and the schema in the DB
#/opt/mssql-tools/bin/sqlcmd -S localhost -U sa -PsaTemp0r@ryP@ssw0rd -d master -i sqlPostInstallScript.sql
for i in `ls ./sqlscripts/* | sort -V`; do
    echo $i && /opt/mssql-tools/bin/sqlcmd -S localhost -U sa -PsaTemp0r@ryP@ssw0rd -i $i ;
done;

# change the Temporary SA Password
/opt/mssql-tools/bin/sqlcmd -S localhost -U sa -PsaTemp0r@ryP@ssw0rd -d master -Q "sp_password NULL, '$1', 'sa'"
```

Compose application

Docker compose

- Tool to define and run multi container Docker applications

Yaml configuration file

- List of services (i.e. containers)
  - Based on image or dockerfile (Build)
- With configuration
  - Environment variables, ports and volume redirection

Run

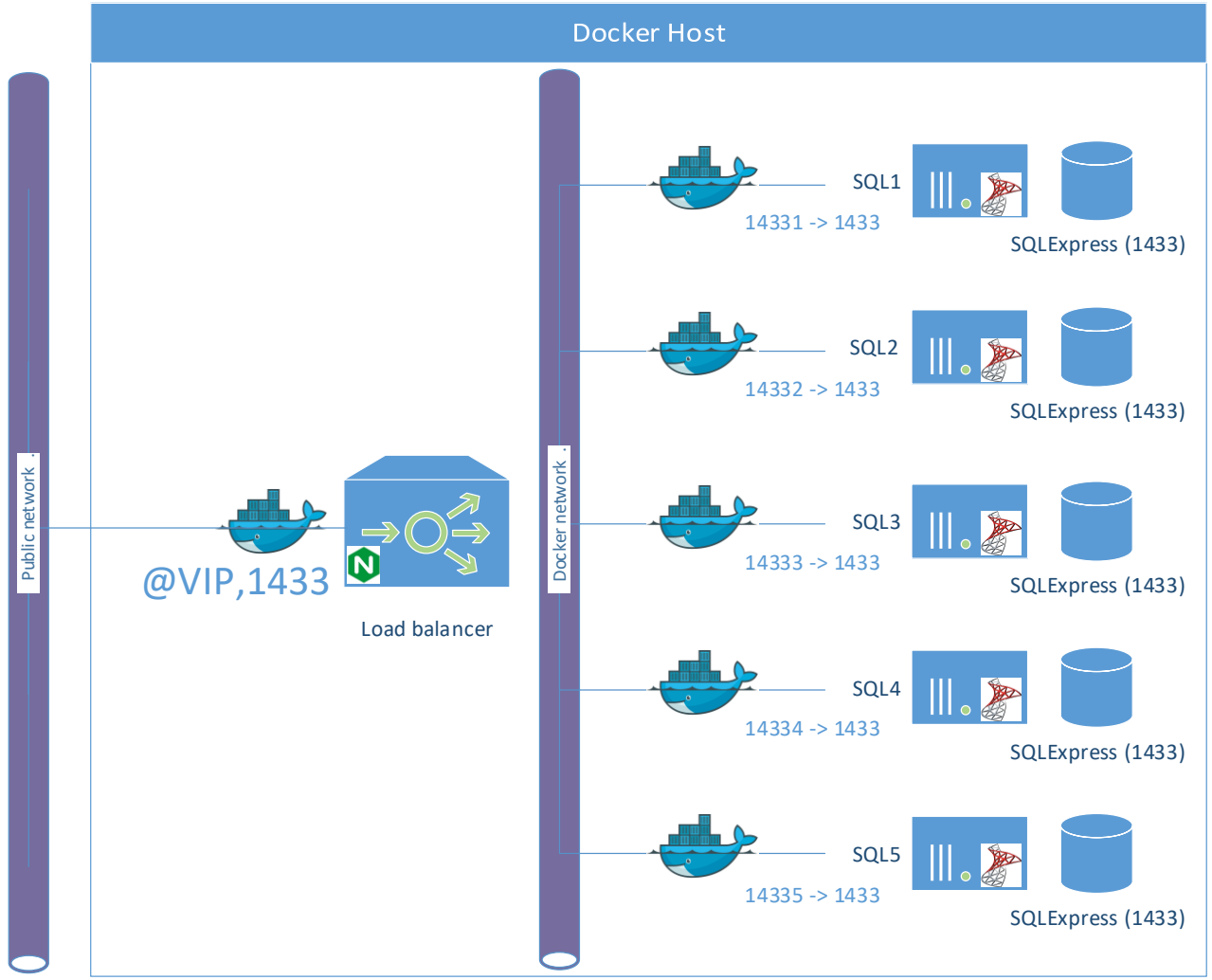
- Docker-compose up
- Docker-compose down

Sample at

https://github.com/dotnet-architecture/eShopOnContainers

Demo – load balance workload

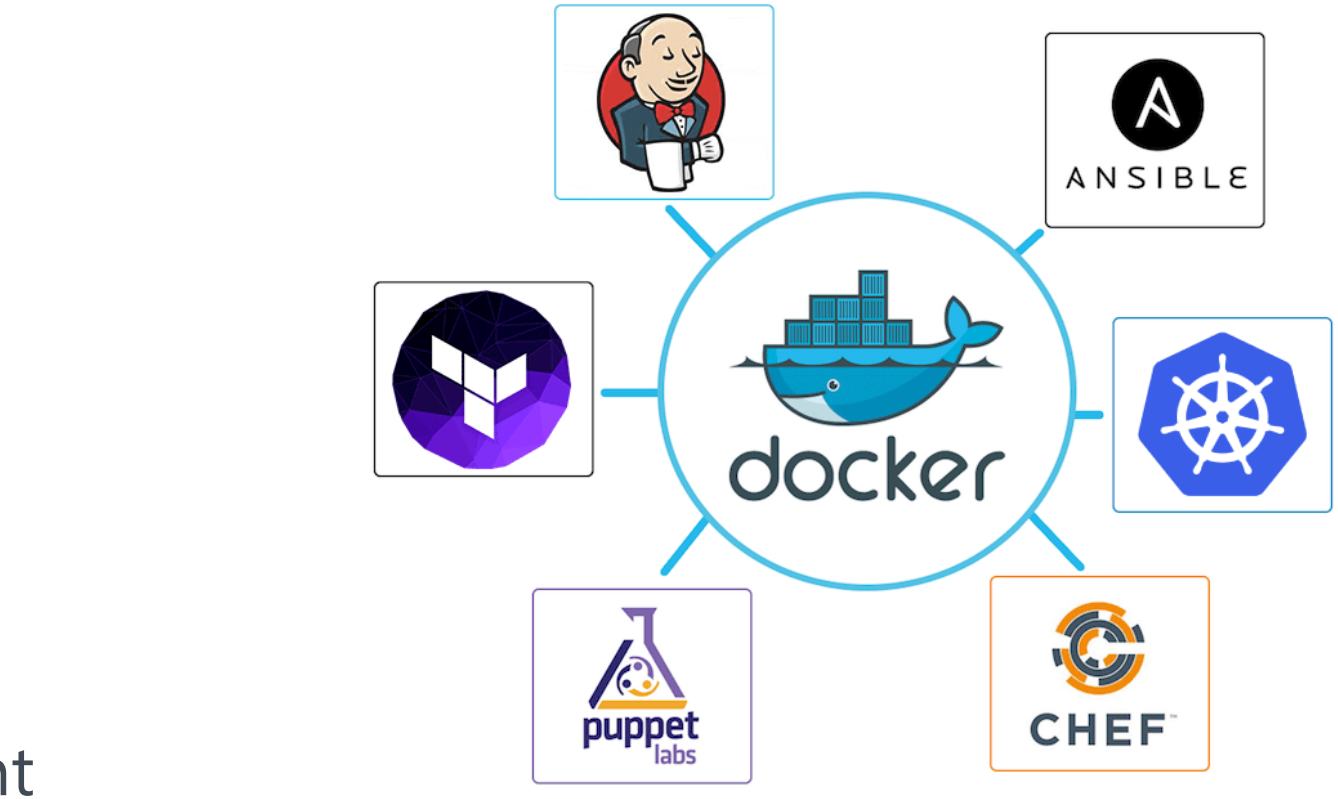
```
services:

  lbsqlnginx:
    image: nginx
    hostname: lbsqlnginx
    container_name: lbsqlnginx
    depends_on:
      - sql1
      - sql2
      - sql3
      - sql4
      - sql5
    ports:
      - "1433:1433"
    volumes:
      - /home/christophe/nginx.conf:/etc/nginx/nginx.conf:ro

  sql1:
    image: microsoft/mssql-server-linux
    hostname: sql1
    container_name: sql1
    ports:
      - "14331:1433"
    environment:
      - MSSQL_SA_PASSWORD=Password1!
      - ACCEPT_EULA=Y
      - MSSQL_PID=express
```

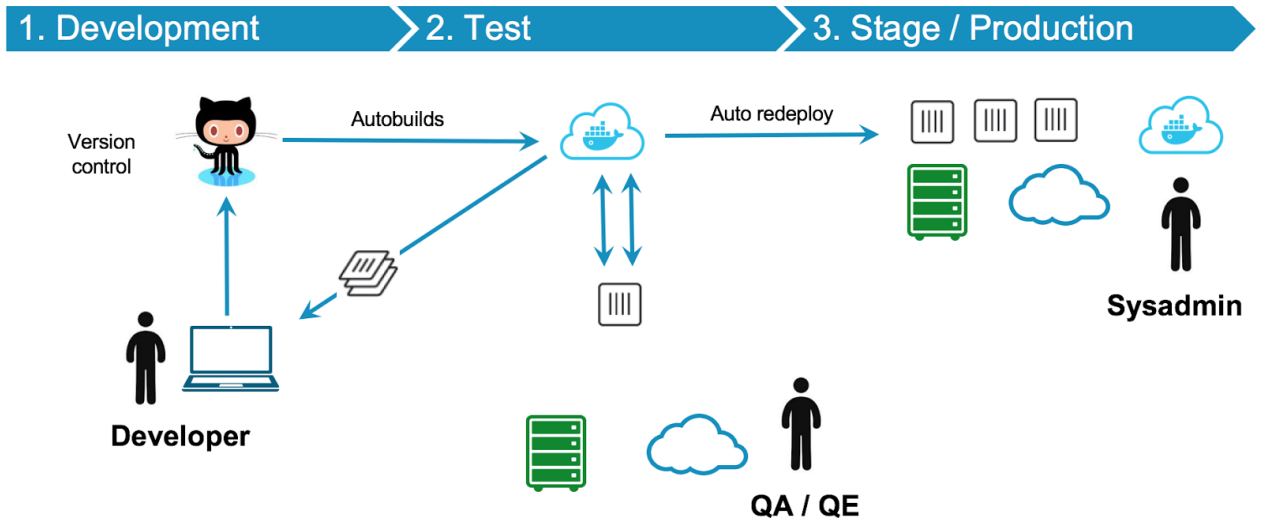


```
root@LinSQLSat735:/home/christophe# docker-compose up -d
Creating network "christophe_default" with the default driver
Creating sql2 ... done
Creating sql3 ... done
Creating sql5 ... done
Creating sql1 ... done
Creating sql4 ... done
Creating lbsqlnginx ... done
root@LinSQLSat735:/home/christophe# docker ps
CONTAINER ID        IMAGE                               COMMAND                  CREATED             STATUS                            PORTS                                 NAMES
2d6d11d2797a        nginx                               "nginx -g 'daemon of…"   6 seconds ago       Up 4 seconds (health: starting)   80/tcp, 0.0.0.0:1433->1433/tcp        lbsqlnginx
d7e7f3dcfecf        microsoft/mssql-server-linux        "/opt/mssql/bin/sqls…"   10 seconds ago      Up 5 seconds (health: starting)   0.0.0.0:14334->1433/tcp               sql4
4401d8bf3b18        microsoft/mssql-server-linux        "/opt/mssql/bin/sqls…"   10 seconds ago      Up 7 seconds (health: starting)   0.0.0.0:14331->1433/tcp               sql1
bb6fb83eab49        microsoft/mssql-server-linux        "/opt/mssql/bin/sqls…"   10 seconds ago      Up 7 seconds (health: starting)   0.0.0.0:14333->1433/tcp               sql3
668a0bcc3db2        microsoft/mssql-server-linux        "/opt/mssql/bin/sqls…"   10 seconds ago      Up 7 seconds (health: starting)   0.0.0.0:14335->1433/tcp               sql5
7e9f0720d4bb        microsoft/mssql-server-linux        "/opt/mssql/bin/sqls…"   10 seconds ago      Up 8 seconds (health: starting)   0.0.0.0:14332->1433/tcp               sql2
root@LinSQLSat735:/home/christophe# sqlcmd -S 127.0.0.1,1433 -U SA -P 'Password1!' -Q "select @@servername"

-----------------------------------------------------------------------------------------------------------------------------
sql1

(1 rows affected)
root@LinSQLSat735:/home/christophe# sqlcmd -S 127.0.0.1,1433 -U SA -P 'Password1!' -Q "select @@servername"

-----------------------------------------------------------------------------------------------------------------------------
sql2

(1 rows affected)
root@LinSQLSat735:/home/christophe# sqlcmd -S 127.0.0.1,1433 -U SA -P 'Password1!' -Q "select @@servername"

-----------------------------------------------------------------------------------------------------------------------------
sql3

(1 rows affected)
root@LinSQLSat735:/home/christophe# sqlcmd -S 127.0.0.1,1433 -U SA -P 'Password1!' -Q "select @@servername"

-----------------------------------------------------------------------------------------------------------------------------
sql4

(1 rows affected)
root@LinSQLSat735:/home/christophe# sqlcmd -S 127.0.0.1,1433 -U SA -P 'Password1!' -Q "select @@servername"

-----------------------------------------------------------------------------------------------------------------------------
sql5

(1 rows affected)
root@LinSQLSat735:/home/christophe# docker-compose down
Stopping lbsqlnginx ... done
Stopping sql4       ... done
Stopping sql1       ... done
Stopping sql3       ... done
Stopping sql5       ... done
Stopping sql2       ... done
Removing lbsqlnginx ... done
Removing sql4       ... done
Removing sql1       ... done
Removing sql3       ... done
Removing sql5       ... done
Removing sql2       ... done
Removing network christophe_default
root@LinSQLSat735:/home/christophe#
```

Next Step

- Orchestration
  - Docker swarm
  - Kubernetes
- Infastructure as code
  - Terraform
  - Cloud formation
- Configuration management
  - Chief
  - Ansible
- CI / CD
  - Jenkins
  - GitHub





Containers 101 – Key points

Small system footprint

Lightweight -> better efficiency on host servers

Single image

Multiple deployments (dev / test / prod)

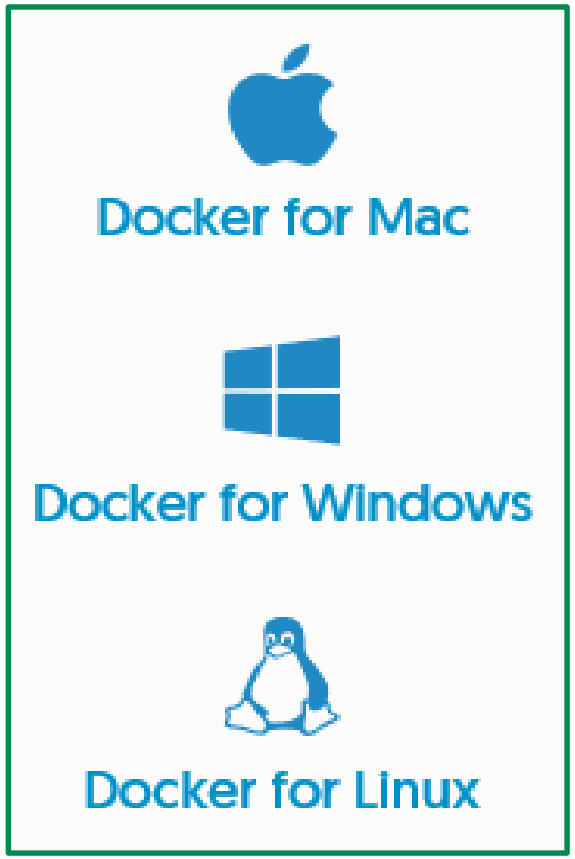
Avoid : “it works on my computer” !

Will always run the same

Regardless of where it is deployed

WORA : Write Once Run Anywhere

PODA : Package Once Deploy Anywhere



Containers 101 – Key points

Virtualization 2.0 :

- Step #1 : hardware virtualization (Hyper-V, VMware)
- Step #2 : OS virtualization (Linux containers, Windows containers, Docker)

Stack

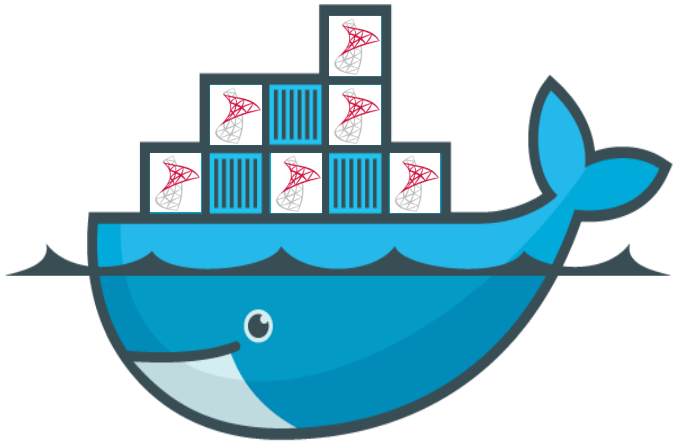
- Each layer can use the libraries provided by the underlaying level
- Each layer can evolve independently from each other

Deployment

- DevOps style
- Accelerate development and application shipment

Strong market trend

- Micro services seems to become a standard



Thank you for attending

Q&A