David Barbarin

from the development to the production
Our sponsors
Agenda

Feedback session ...

- Development project with SQL Server containers
- Production environment with SQL Server containers
Development project with SQL Server containers
Development project with SQL Server containers

History of the project: DMK tool

> Think about Ola Hallengren project but starting on 2012 ...
History of the project: DMK tool

- Think about Ola Hallengren project but starting on 2012 ...
- Primarily created to handle AlwaysOn AG (to get rid of false positive issues)
  - Very simple set of scripts but no versioning process
  - No support for versions prior 2012
  - SSMS as development tool
History of the project: DMK tool

> Think about Ola Hallengren project but starting on 2012 ...
> Primarily created to handle AlwaysOn AG (to get rid of false positive issues)
  > Very simple set of scripts but no versioning process
  > No support for versions prior 2012
  > SSMS as development tool
> From 2014 to 2016 ...
  > Scripts part of a true software release management cycle
  > Minimal versioning with SVN
  > SSMS --> SSDT (database project & state-oriented project)
  > Supported versions -->2012 – 2016
  > Shared environment with different SQL Server instance versions + topologies
Development project with SQL Server containers

By 2017 ... time to make a first observation
Development project with SQL Server containers

By 2017 ... time to make a first observation

> Most of our customers are using this tool
> More features to support over with new versions 2008 – 2017 Win and Linux
> Time-to-market and service quality is impacted 😞
Development project with SQL Server containers

By 2017 … time to make a first observation

> Most of our customers are using this tool
> More features to support over with new versions 2008 – 2017 Win and Linux)
> Time-to-market and service quality is impacted 😞
> New challenges to address
Development project with SQL Server containers

By 2017 ... time to make a first observation

> Most of our customers are using this tool
> More features to support over with new versions 2008 – 2017 Win and Linux
> **Time-to-market and service quality is impacted 😞**
> **New challenges to address**
> Code complexity increased -> Changes to be managed carefully to avoid regressions
> SQL Server release pace increased -> Need to introduce more resources to keep up
> Testing phase increased -> Manual process (bugs / regressions)
> Dev environment doesn’t fit with new context
  > Shareable = no work isolation for developer team + inconsistency +
  > Static = inconsistency + queuing + long and costly maintenance times
Development project with SQL Server containers

DevOps as a new trend at customer shops ...

... and fits with our context
Development project with SQL Server containers

DevOps as a new trend at customer shops … … and fits with our context

DevOps pillars

> Process
Development project with SQL Server containers

DevOps as a new trend at customer shops ...

... and fits with our context

DevOps pillars

> Process

> Source control
Development project with SQL Server containers

DevOps as a new trend at customer shops ...

... and fits with our context

DevOps pillars

> Process
> Source control
> Unit testing
Development project with SQL Server containers

DevOps as a new trend at customer shops ...

... and fits with our context

DevOps pillars

> Process
> Source control
> Unit testing
> Infrastructure as code
Development project with SQL Server containers

DevOps as a new trend at customer shops ...

... and fits with our context

DevOps pillars

- Process
- Source control
- Unit testing
- Infrastructure as code
- Continuous integration/delivery
- Continuous monitoring
Development project with SQL Server containers

Using Docker was the first and obvious way ...
Development project with SQL Server containers

Using Docker was the first and obvious way ...

- Faster deployment & lower resource footprint
- Consistency between tests and stages
- Isolated environments for developers
Development project with SQL Server containers

Using Docker was the first and obvious way ...

- Faster deployment & lower resource footprint
- Consistency between tests and stages
- Isolated environments for developers

- Managing **Docker Windows containers** for MSSQL is challenging
  - “From scratch” process -> Time consuming + security concerns
  - Need to introduce an internal registry in the architecture
- Storage footprint --> (10GB per image + 100GB DB size) x N developers ??
- Security features like TDE not straightforward to implement
Development project with SQL Server containers

Windocks for Docker Windows containers
Development project with SQL Server containers

CI architecture (current implementation)

- Unit tests
- Integration tests
- Copy DBs + scripts
- Restore DBs
- Copy DBs to build images
- Backup files
Development project with SQL Server containers

First achieved objectives

- Bugs and regressions fixed in early stage
- Refocusing on developing features
- Time-to-market reduced (monthly releases)

- To introduce code coverage
- To improve security of CI pipeline
- To address storage issue on Linux side
- To integrate database scripts to versioning
Production environment with SQL Server containers
Production project with SQL Server containers

Customer case

Docker Engine standalone

NFS
Production project with SQL Server containers

Customer case

Docker Engine standalone

NFS
Production project with SQL Server containers

Customer case

Initial scenario

- Customer moved from Oracle Linux to SQL Server Linux
- Strong Linux background culture / Docker enthusiastic
- openSUSE Leap 42.3
- Docker 17.09.1-ce --> 4 CPUS / 25 GB RAM / 10Gbits Net
- 3 MSSQL containers
  - SQL Server 2017 CTP image !!
  - NFS 3.1 but 4.2+ required for MSSQL !! -> fallocate() issues
  - No configuration / performance setting applied
  - Some high-critical applications --> No HA !!
  - No CPU / Memory / IO-bound applications
Production project with SQL Server containers

Customer case

Production leads to different concerns:

> Product’s maturity ???
> Persistent storage ?? (blocker for a long time)
> Docker CE vs EE (UCP, DTR, security, built-in monitoring ..)
> MSSQL Docker images “production ready” by default?
  > Configuration best practices / management / monitoring
  > Security (secrets/access isolation/networking/trusted images …)
> Management / Patching / Backup / Restore / Monitoring …
> HA / Load balancing?
  > Orchestrators (Docker Swarm / Kubernetes / OpenShift …)
> No support for SQL FCI / AG !!
> New deployment model => stacks / services
> Health check routines are important!
Production project with SQL Server containers

Customer case

Production leads to different concerns:
>
> Product’s maturity ???
> Persistent storage ?? (blocker for a long time)
> Docker CE vs EE (UCP, DTR, security, built-in monitoring ..)
> MSSQL Docker images “production ready” by default?
> Configuration best practices / management / monitoring
> Security (secrets/access isolation/networking/trusted images …)
> Management / Patching / Backup / Restore / Monitoring …
> HA / Load balancing?
> Orchestrators (Docker Swarm / Kubernetes / OpenShift …)
> No support for SQL FCI / AG !!
> New deployment model => stacks / services
> Health check routines are important!
Production project with SQL Server containers

Customer case

Production leads to different concerns:

> Product’s maturity ???
> Persistent storage ?? (blocker for a long time)
> Docker CE vs EE (UCP, DTR, security, built-in monitoring ..)
> MSSQL Docker images “production ready” by default?
  > Configuration best practices / management / monitoring
  > Security (secrets/access isolation/networking/trusted images ...)
> Management / Patching / Backup / Restore / Monitoring ...
> HA / Load balancing?
  > Orchestrators (Docker Swarm / Kubernetes / OpenShift ...)
  > No support for SQL FCI / AG !!!!
> New deployment model => stacks / services
  > Health check routines are important!
Production project with SQL Server containers

Customer case

Production leads to different concerns:
> Product’s maturity ???
> Persistent storage ?? (blocker for a long time)
> Docker CE vs EE (UCP, DTR, security, built-in monitoring ..)
> MSSQL Docker images “production ready” by default?
  > Configuration best practices / management / monitoring
  > Security (secrets/access isolation/networking/trusted images …)
> Management / Patching / Backup / Restore / Monitoring …
> HA / Load balancing?
  > Orchestrators (Docker Swarm / Kubernetes / OpenShift …)
  > No support for SQL FCI / AG !!
> New deployment model => stacks / services
> Health check routines are important!
Production project with SQL Server containers

Customer case

Production leads to different concerns:

> Product’s maturity ???
> Persistent storage ?? (blocker for a long time)
> Docker CE vs EE (UCP, DTR, security, built-in monitoring ...)
> MSSQL Docker images “production ready” by default?
  > Configuration best practices / management / monitoring
  > Security (secrets/access isolation/networking/trusted images ...)
> Management / Patching / Backup / Restore / Monitoring ...
> HA / Load balancing?
  > Orchestrators (Docker Swarm / Kubernetes / OpenShift ...)
  > No support for SQL FCI / AG !!
> New deployment model => stacks / services
> Health check routines are important!
Production project with SQL Server containers

Customer case

Production leads to different concerns:

> Product’s maturity ???
> Persistent storage ?? (blocker for a long time)
> Docker CE vs EE (UCP, DTR, security, built-in monitoring ...)
> MSSQL Docker images “production ready” by default?
> Configuration best practices / management / monitoring
> Security (secrets/access isolation/networking/trusted images ...)
> Management / Patching / Backup / Restore / Monitoring ...
> HA / Load balancing?
> Orchestrators (Docker Swarm / Kubernetes / OpenShift ...)
> No support for SQL FCI / AG !!
> New deployment model => stacks / services
> Health check routines are important!
Production project with SQL Server containers

Customer case – Management

**Based image**

- `> FROM microsoft/mssql-server-linux:<tag>`
- `> mssql-cli, htop`

**Installation**

- `/opt/mssql/bin/` (binaries)
- `/var/opt/mssql/` (db files, tlog files, logs, secrets ...)
- `> disk = /u[0-9]*`
- `tempdb configuration`
- `Memory capping`
- `Trace flags`

- OS performance settings
  - Kernel settings
  - Disks settings
  - Swap files

**Management tools?**

**Configuration Best practices?**

**Maintenance**

- `> DMK maintenance`

**Monitoring**

- `> SQL Server Built-in tools – alerts, jobs ...`
- `> Third-party tool => InfluxDB, Grafana, Kankuru ...`

**Performance Best practices?**

**Database backups?**

**Index, stats, db check integrity**

**Security**

- `> Prevent using sa`
- `> Special user for apps`
Production project with SQL Server containers

Customer case - Performance

Network

- Choice of network may be important (Host vs Bridge vs overlay vs Calico)

Storage

- NFS performance fits in our context (no diff between host machines and containers)
- Docker plugins from storage providers like PureStorage, Nutanix etc ...
- CSI is available for K8s, coming soon for Docker Swarm

CPU / Memory

- Capping either from Docker itself or from SQL Server settings (max server memory, processor affinity)
Production project with SQL Server containers

Customer case
Production project with SQL Server containers

Customer case

Docker Engine

NFS

Docker Engine

NFS

Docker Engine
Production project with SQL Server containers

Customer case
Production project with SQL Server containers

Customer case
Production project with SQL Server containers

Customer case
Production project with SQL Server containers

Customer case

Docker Engine

NFS

HA Proxy

Nagios

Docker Engine

NFS

Docker Engine

NFS

Docker Engine
Any questions?
Thank you for attending!