Containerized Delivery from the trenches with Visual Studio, Azure and Docker

Cornell Knulst

Xpirit

Cornell Knulst

Cloud Architect/ALM Consultant

Xpirit Netherlands

@CornellKnulst

Xpirit

Content

- Why containers?
- #1 Impact and limitations
- #2 Importance of container layering
- #3 Containerization and ALM
- #4 Secure your containerized workload
- *#5 Monitoring*
- Benefits and drawbacks

Why containers?

The cloud has changed expectations

Availability

100% Uptime

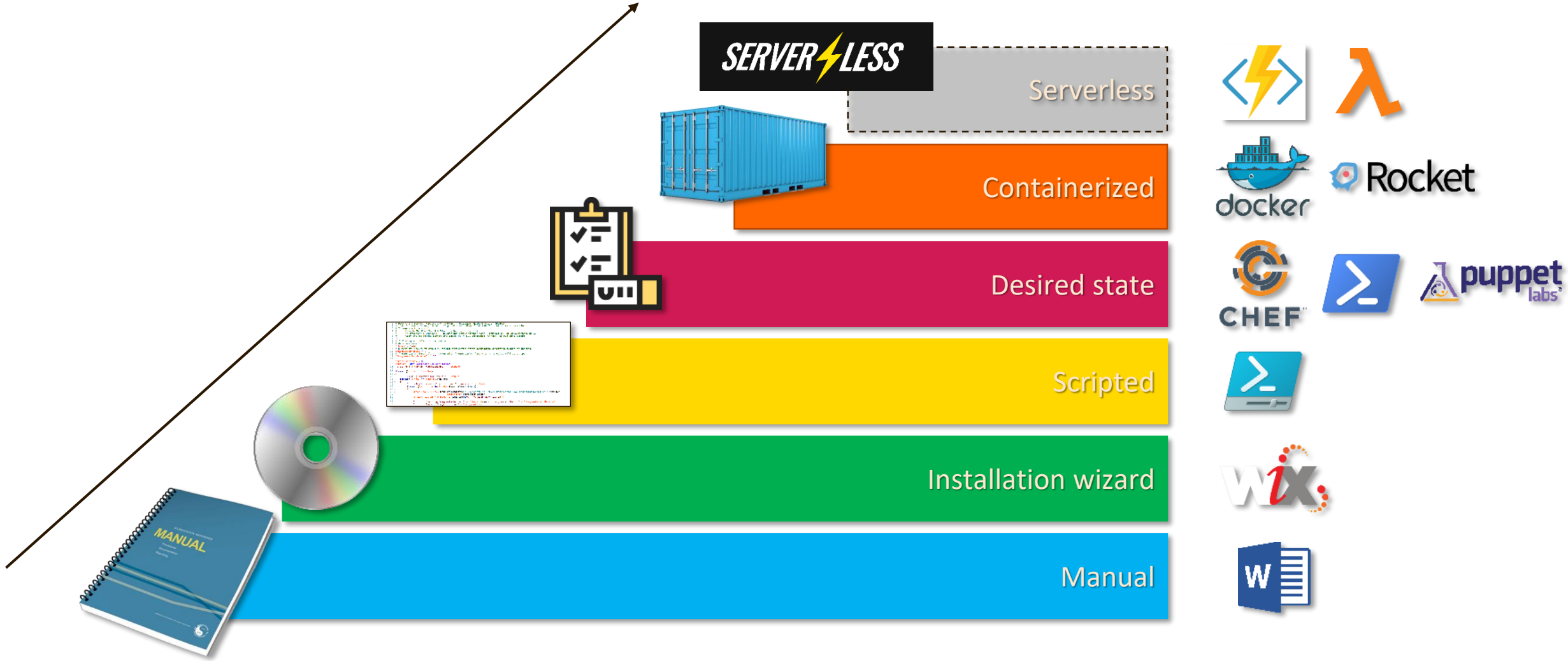
Hyper-scale

From startup to enterprise

Agility

Deliver just in time speed

Evolution of Application Delivery

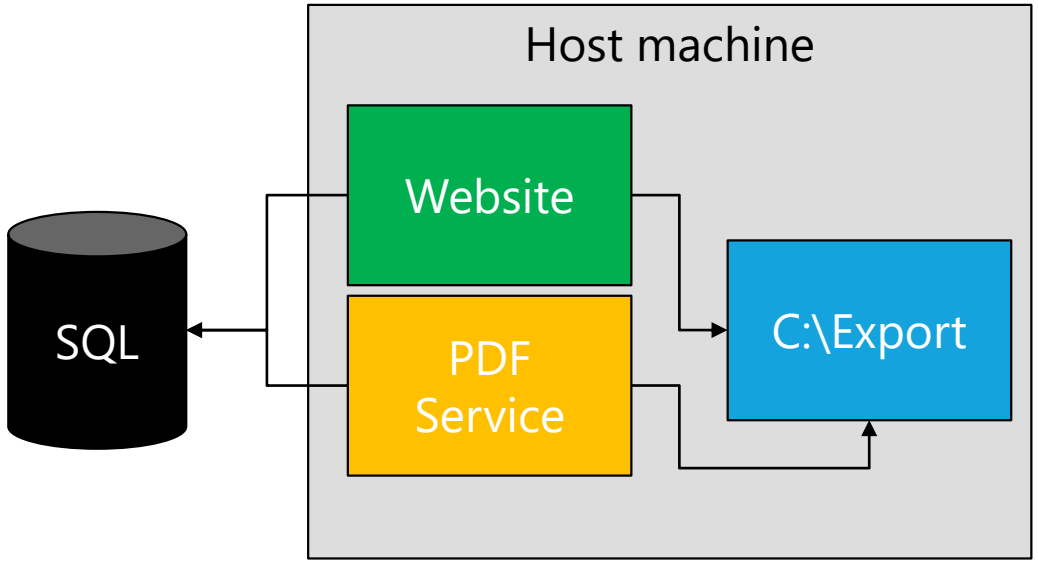


Why containers in a serverless world?

Two customer cases as an example..

Containers as the way forward- Full.NET Case

- Financial Dashboards
  - Fortune 500
  - Rollup Oracle, SAP, …
- .NET 4.X
- Current situation
  - On-premise WiX installers
  - Installations done by customers
  - Lot of different code lines
- Demand:
  - SaaS cloud offering
  - Prepared for Oracle, SAP cloud offerings

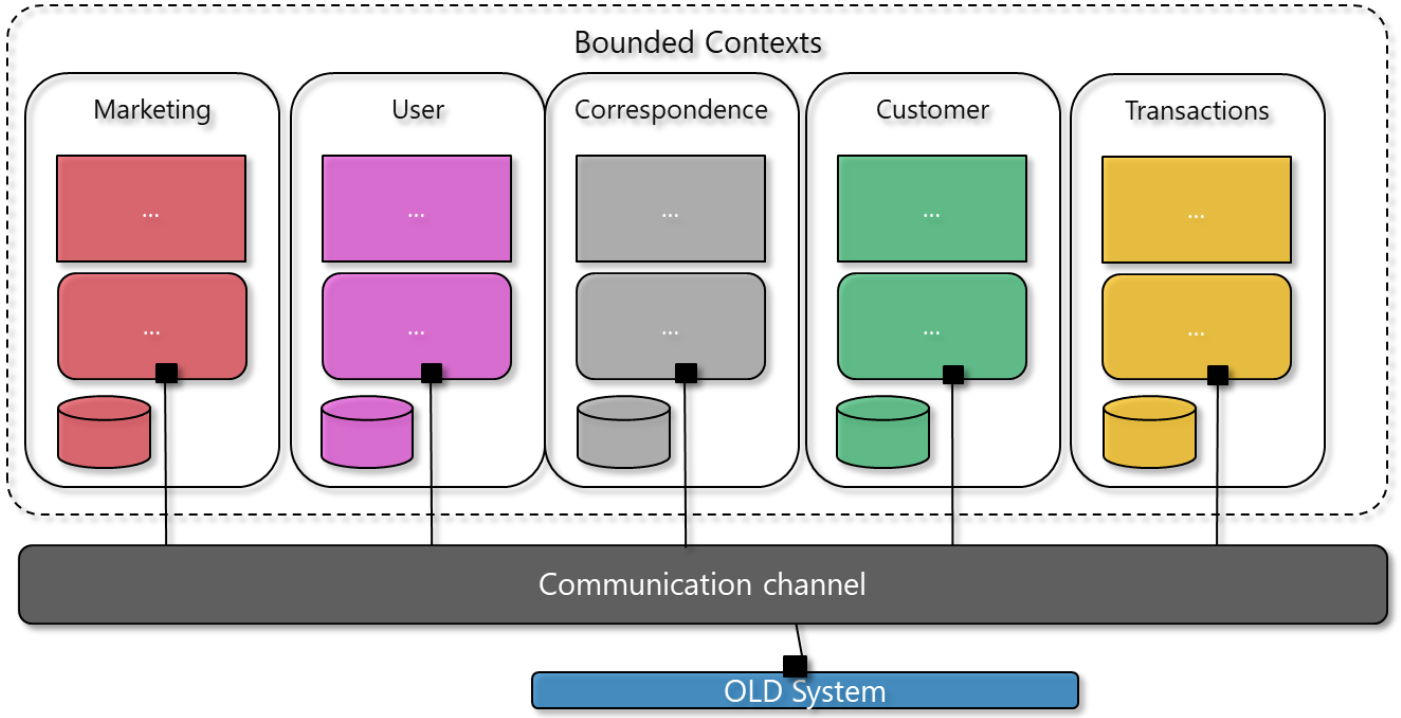


Why containers in a serverless world

- Serverless is not an option
  - Current application stack
  - Delivery on both cloud and on-premise
  - Phased transition of customers to the cloud
- Possibility to move workload to AWS

Containers as the way forward- .NET Core Case

- New financial platform:
  - Payment Service Provider
  - Bank account implementation
- .NET Core 2.0
- Unique selling point
  - Europe only
  - Always/high availability
  - Competitive pricing



Why containers in a serverless world

- Movement of workload over Azure and AWS
  - No vender lock-in
  - Higher availability in Europe
  - Competitive pricing
- Learning curve in serverless architecture

Containerized vs Serverless

Containerized

- Pay-per-resource
- Planned scaling
- *On-premise and cloud*
- *Vendor independent*
- Management of infrastructure
- Stateless per container lifecycle
- *Brownfield scenario's*

Serverless

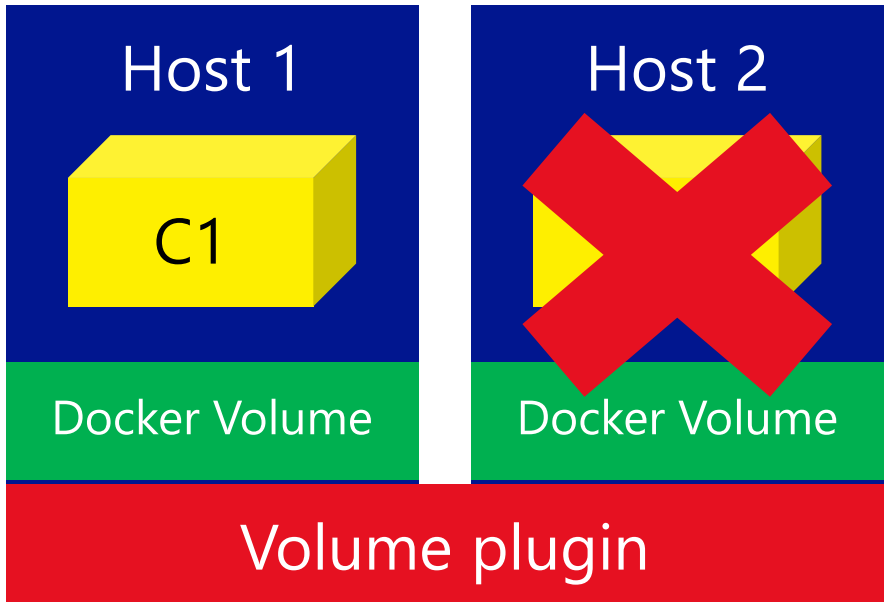
- Pay-per-use
- Automatic scaling
- Cloud-native
- Vendor locked
- Infrastructure managed by vendor
- Stateless per call

ACI: the best of both worlds…

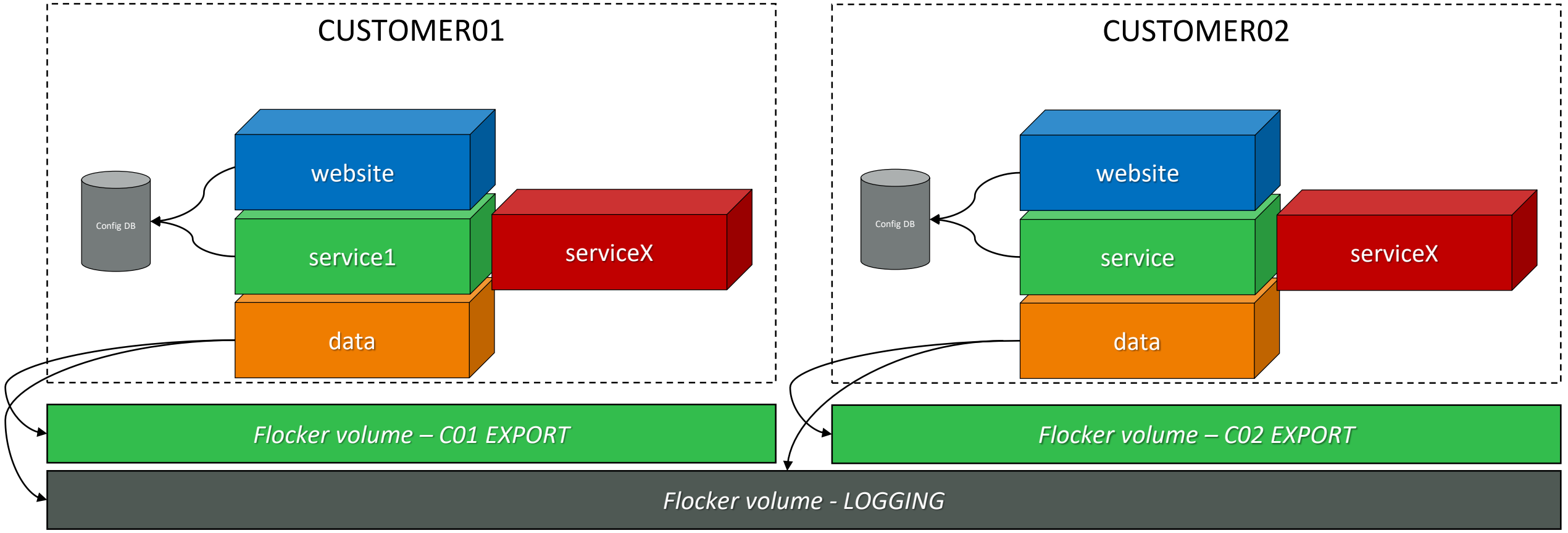
#1 Impact and limitations of containerizing Windows applications

Container characteristics

- Isolation
  - OS virtualization
  - Selective scaling
  - Docker Volumes to share data across containers
- Immutable
  - Reproducible
  - Pets vs cattle
- Stateless
  - Scaling
  - Statefull during container lifecycle
  - Volume plugins, external storage to persist state



Example of impact of those characteristics

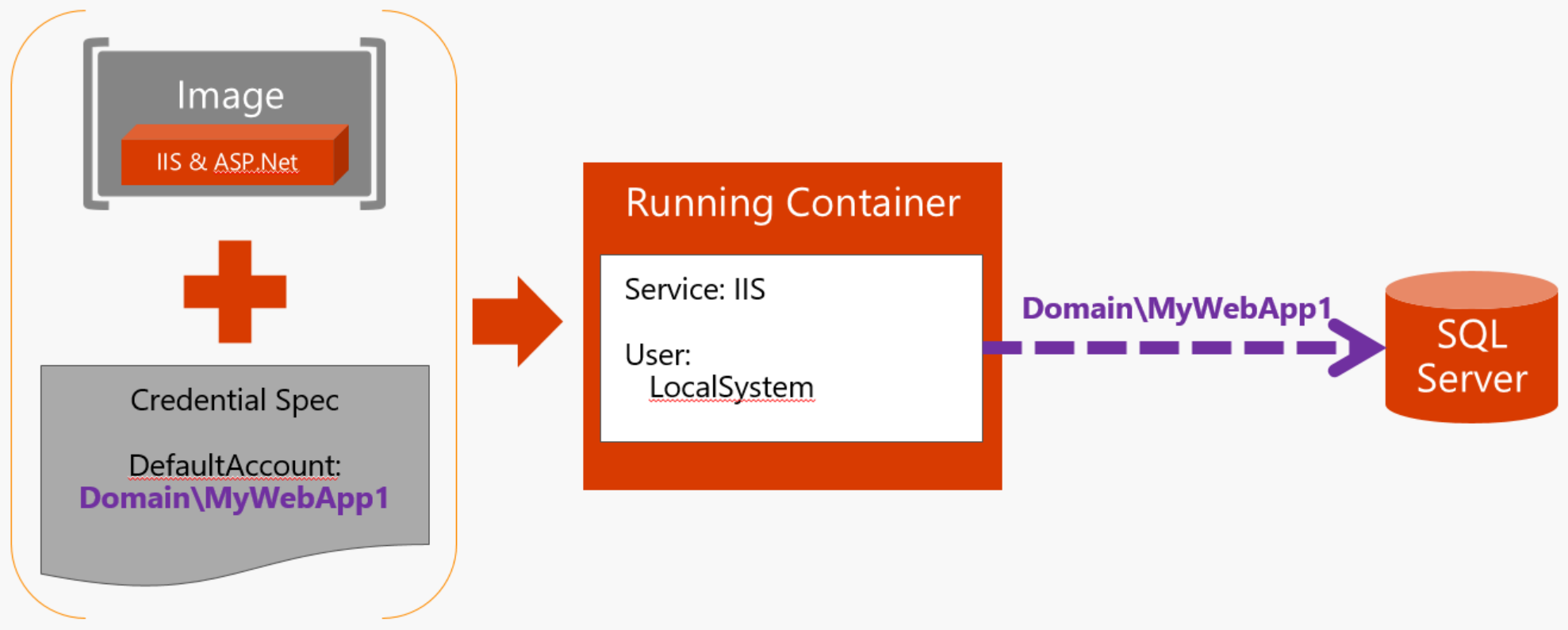


Limitations

- Windows Server 2016
  - Even Windows 10 uses a WS2016 distribution (Hyper-V)
- Foreground processes (ServiceMonitor.exe)

Limitations

- Containers cannout be domain joined
  - gMSA http://bit.ly/2tWGloy



Limitations

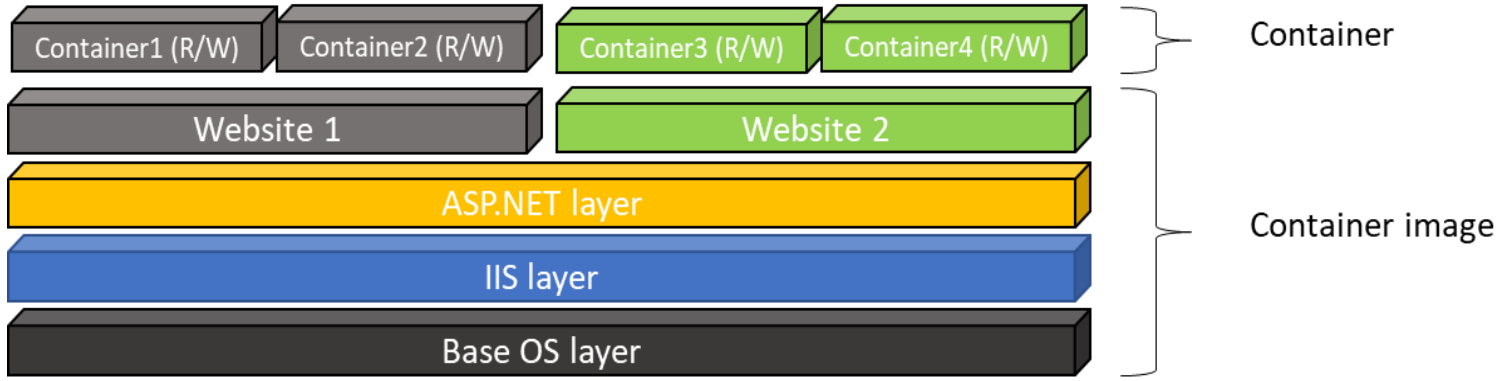
- ENV variables instead of config files

```
string configuredConnectionString = Environment.GetEnvironmentVariable(NAME);
if (string.IsNullOrWhiteSpace(configuredConnectionString))
{
        ConnectionStringSettings connectionStringInConfig = ConfigurationManager.ConnectionStrings[NAME];

        if (connectionStringInConfig == null || string.IsNullOrWhiteSpace(connectionStringInConfig.ConnectionString))
                throw new Exception("Unable to find connection string");

        configuredConnectionString = connectionStringInConfig.ConnectionString;
}
```

#2 Importance of container layering

Image layering

- Keep layers small
- Sequence matters
- Combine actions
- Multi-staged builds



#3 Containerization and ALM

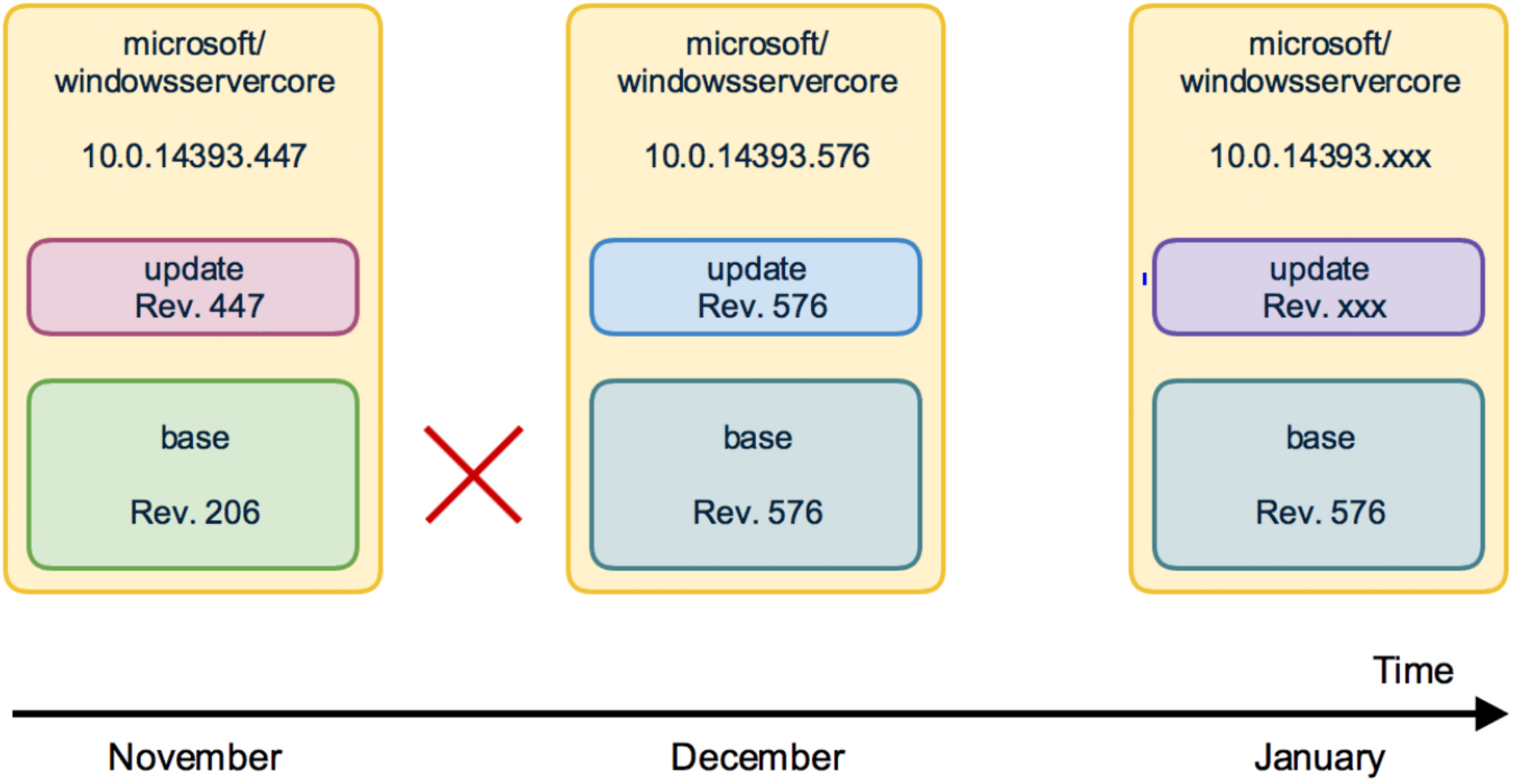
Selecting the right Container Registry…

- Container registry (Tool, location, availability, costs)

Registry type	Public/ private	On-premise/ cloud	Support for other package management technologies	License and security vulnerability scanning	Technical maintenance	Costs
Nexus Repository OSS	Private	On-premise	Maven, Docker, NuGet, Npm, PyPI, Bower,..	No	Client (no support from Nexus side)	Free
Nexus Repository Pro	Private	On-premise	Maven, Docker, NuGet, Npm, PyPI, Bower,..	For some technologies.	Client (support from Nexus side)	$10 per user/per month.
Artifactory	Private	On-premise	Docker, NuGet, Npm, PyPI, Bower,PHP, Vagrant, ..	Via Black Duck add-in. Support for Java (using Maven, Ivy or Gradle) and NuGet artifacts.	Client	$2,950 (without security scan) $14,400/year
Docker Trusted Registry	Private	On-premise	No support	Docker Security Scanning	Client	$150 per month
Docker Hub registry	Public	Cloud	No support	Docker Security Scanning Add-On.	Docker	--
Docker Trusted Registry	Private	Cloud	No support	Docker Security Scanning Add-On	Docker	$7 per month^ (Docker Security Scanning included)
Azure Container Registry	Private	Cloud	No support	Via integration with Aqua, and TwistLock.	Microsoft	TwistLock (free dev edition) $64 per month*
AWS Container Registry	Private	Cloud	No support	Via integration with Aqua, and TwistLock.	AWS	TwistLock (free dev edition) $64 per month*
Google Container Registry	Private	Cloud	No support	Via integration with Aqua, and TwistLock.	Google	TwistLock (free dev edition) $92 per month*

Windows updates…

docker pull microsoft/nanoserver

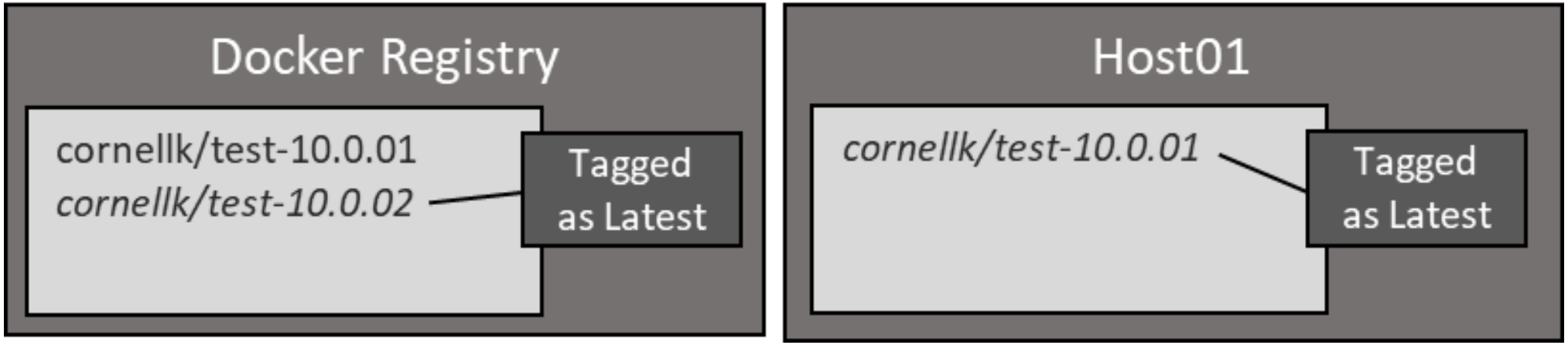


Explicit dependency management

- Dockerfile without version tag

```
FROM    microsoft/windowsservercore:10.0.14393.1770
SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop';"]
RUN Add-WindowsFeature NET-Framework-45-Core,NET-Framework-45-ASPNET
```

ng tag!

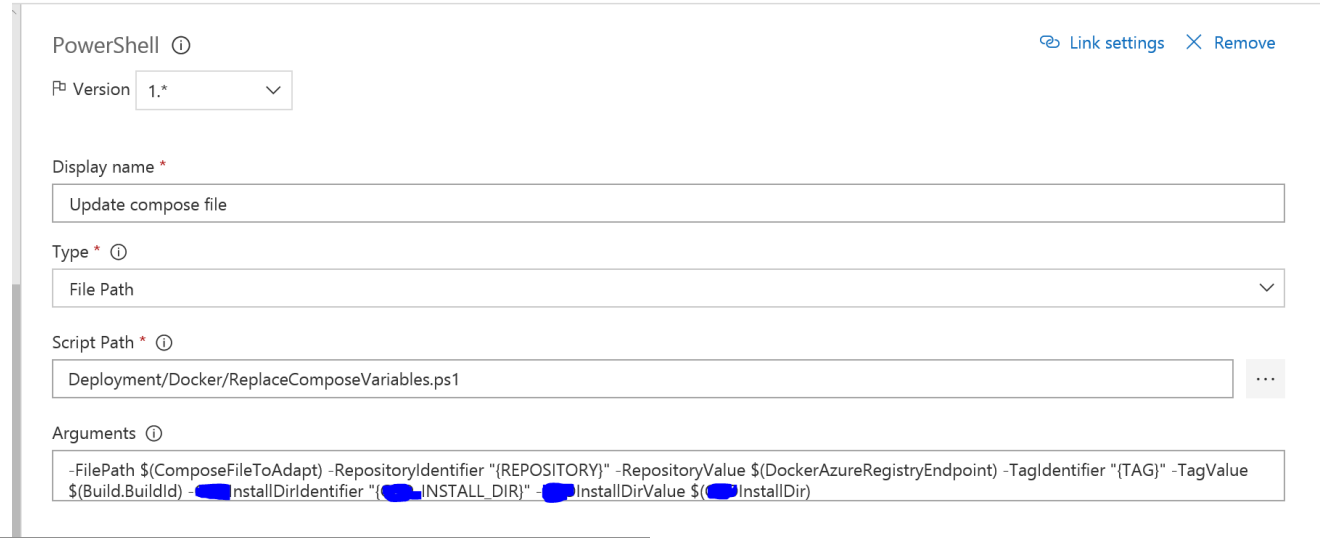


docker run -it --name cornellk/test cmd

Explicit dependency management - Compose

- Replace version of image in deployment files during build

```
version: '2'
services:
  data:
    image: "{REPOSITORY}/data:latest"
    volumes:
      - 'loggingServiceX:{INSTALL_DIR}/ServiceX/Logs'
      - 'loggingServiceY:{INSTALL_DIR}/ServiceY/Logs'
      - 'loggingWebsite:{INSTALL_DIR}/Website/Logs'
      - 'export:{INSTALL_DIR}/ServiceY/Files'

  website:
    image: "{REPOSITORY}/website:{TAG}"
    environment:
      - CONNECTIONSTRING
    volumes_from:
      - data
    ports:
      - "80:80"

  servicex:
    image: "{REPOSITORY}/servicex:{TAG}"
    environment:
      - CONNECTIONSTRING
    volumes_from:
      - data

  servicey:
    image: "{REPOSITORY}/servicey:{TAG}"
    environment:
      - CONNECTIONSTRING
    volumes_from:
      - data
```

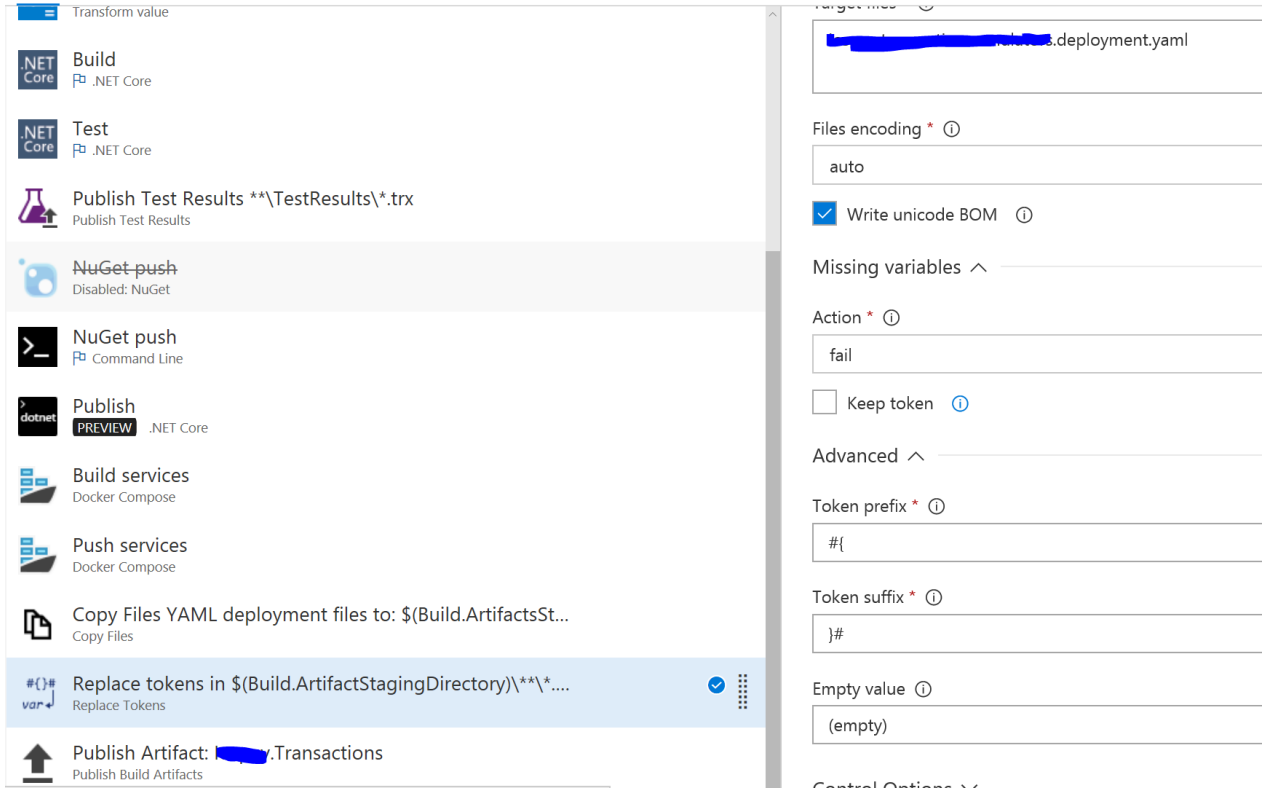



```
Param(
    [Parameter(Mandatory=$true)][string]$FilePath,
    [Parameter(Mandatory=$true)][string]$RepositoryIdentifier,
    [Parameter(Mandatory=$true)][string]$RepositoryValue,
    [Parameter(Mandatory=$true)][string]$TagIdentifier,
    [Parameter(Mandatory=$true)][string]$TagValue,
    [Parameter(Mandatory=$true)][string]$InstallDirIdentifier,
    [Parameter(Mandatory=$true)][string]$InstallDirValue
)
$content = Get-Content $FilePath
$content = $content.replace($RepositoryIdentifier, $RepositoryValue)
$content = $content.replace($TagIdentifier, $TagValue)
$content = $content.replace($InstallDirIdentifier, $InstallDirValue)
$content | Set-Content $FilePath
```

Explicit dependency management – K8s

- Replace version of image in deployment files during build

```
76  ---
77  apiVersion: extensions/v1beta1
78  kind: Deployment
79  metadata:
80    creationTimestamp: null
81    labels:
82      io.kompose.service: [redacted].transactions.gspayment
83    name: [redacted].transactions.gspayment
84  spec:
85    replicas: 1
86    strategy:
87      type: RollingUpdate
88    template:
89      metadata:
90        creationTimestamp: null
91        labels:
92          io.kompose.service: [redacted]transactions.gspayment
93      spec:
94        imagePullSecrets:
95              - name: acrconnection
96        "nodeSelector": {
97          "beta.kubernetes.io/os": "windows"
98        }
99        containers:
100       - env:
101         - name: ASPNETCORE_ENVIRONMENT
102           value: **{ASPNETCORE_ENVIRONMENT}**
103         - name: ApplicationInsights__InstrumentationKey
104           value: **{ApplicationInsights__InstrumentationKey}**
105         - name: BSPayment__Url
106           value: **{BSPayment__Url}**
107       image: [redacted].azurecr.io/[redacted].transactions.gspayment:#{Build.BuildNumber}#
108       name: [redacted]-transactions-gspayment
109       ports:
110       - containerPort: 80
111       resources: {}
112     restartPolicy: Always
113 status: {}
114 ---
```



Working with a cluster

- Technology
  - Swarm vs Kubernetes vs DC/OS
  - Specific features and naming (e.g. pods, secrets, namespaces, deployment files)
- Proxy
  - K8s Ingress, NGINX
- Patch management
  - Immutable infra
  - Azure Container Instances ☺
- Track disk space
  - Clean up unused containers and images (docker ps –a, docker images -a)

#4 Secure your containerized workload

Secure your containerized workload

- Know the origin and content of your images
  - **Content:** Winspector
  - **Authenticity** of images: Docker Notary
  - **Vulnerability scanner:** Docker Security Scan, Aqua, Twistlock
  - Or create your own base images

yment / Docker / webcore / **Dockerfile**

Contents History Compare Blame Edit Rename

```
1  FROM [redacted]/connectivitycore
2  MAINTAINER [redacted] <info@[redacted].com>
3
4  SHELL ["powershell", "-Command", "$ErrorActionPreference = 'Stop';"]
5
6  # Enable web related Windows features
7  RUN Add-WindowsFeature Web-Static-Content,Web-Http-Logging,Web-Stat-Compression,Web-Dyn-Compression,Web-Net-Ext45,Web-Asp-Net45,Web-ISAPI-Ext,Web-ISAPI-Filter,Web-WebSockets,NET-Framework-45-ASPNET ;
8      Enable-WindowsOptionalFeature -FeatureName IIS-ASPNET45,IIS-HttpCompressionDynamic,IIS-HttpCompressionStatic,IIS-HttpLogging,IIS-StaticContent,IIS-WebServer,IIS-WebSockets -Online -All
9
```

Secure your containerized workload

- Harden your images, containers, daemons and hosts
  - Docker benchmark (https://www.cisecurity.org/benchmark/docker/)
  - TLS secure your docker host (see docker run command below)
  - Disable features and services that aren't needed
  - Firewall

```
docker run --rm `
  -e SERVER_NAME=$(hostname) `
  -e IP_ADDRESSES=127.0.0.1,192.168.254.123 `
  -v "C:\ProgramData\docker:C:\ProgramData\docker" `
  -v "$env:USERPROFILE\.docker:C:\Users\ContainerAdministrator\.docker" `
  stefanscherer/dockertls-windows
```

Secure your containerized workload

- Keep secrets out of your container!
  - ~~Environment Variables~~
  - *Docker Volumes*
  - **Cluster Secrets**
- Initialize them from the release pipeline

docker **secret create** [NAME] [VALUE]

docker **service create** --name my-iis -p 8000:8000 --secret src=[VALUE],target="\inetpub\wwwroot\index.html" microsoft/iis:nanoserver

Example secret consuming K8s

kubectl **create secret** *docker-registry* acrconnection --docker-server=[LOCATION] --docker-username=[USERNAME] --docker-password=[PW] --docker-email=[EMAIL]

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    io.kompose.service: [redacted].iedemo
  name: [redacted].iedemo
spec:
  replicas: 1
  strategy:
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        io.kompose.service: [redacted].iedemo
    spec:
      imagePullSecrets:
        - name: acrconnection
      "nodeSelector": {
       "beta.kubernetes.io/os": "windows"
      }
```

#5 Monitoring

Stay tuned…

Benefits and drawbacks

Benefits and drawbacks

- ✓ Better, cheaper, faster deployments
- ✓ Scalable application stack

- ➢ Learning curve
  - Concepts and technology
  - Managing secrets
- ➢ Management of container cluster
  - Future: Azure Container Instances

Thank You!!

@cornellknulst

https://www.solidalm.com

Coming soon:

https://www.containersonwindows.com

Xpirit