MIRANTIS

Get Your Windows Apps Ready for Kubernetes

Steven Follis
Mirantis

CLOUD NATIVE COMPUTING FOUNDATION WEBINAR SERIES

Agenda

Why Windows Containers?

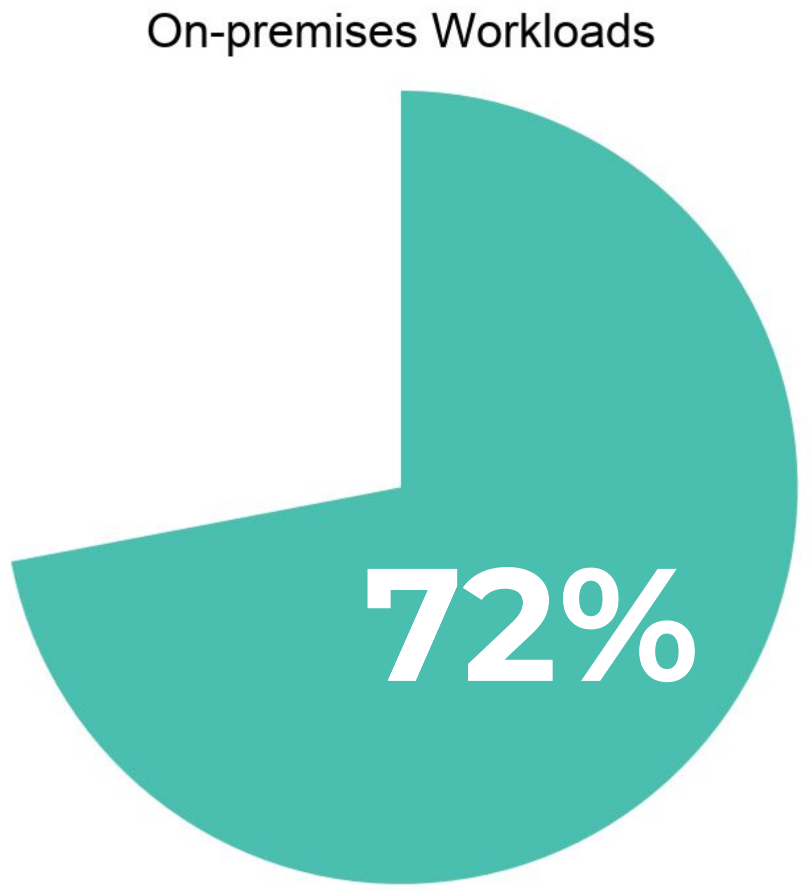
Use Cases

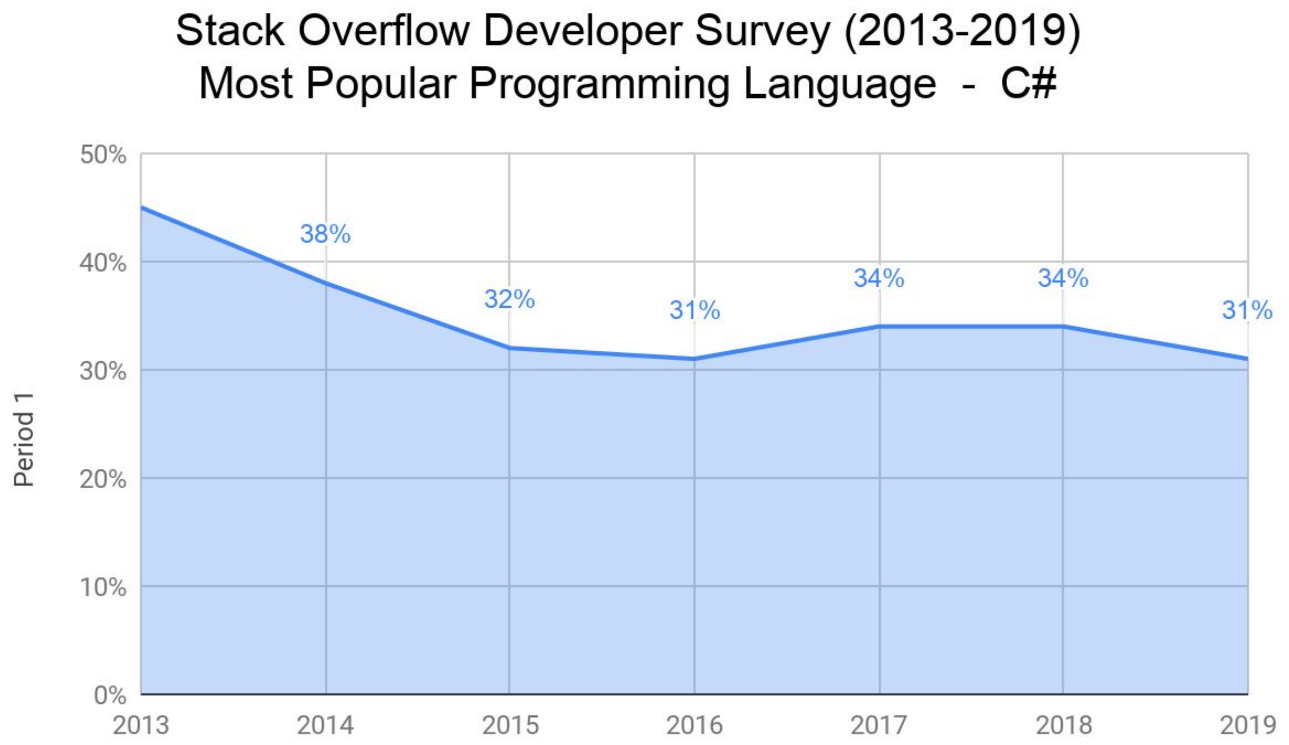
Real-world Considerations

MIRANTIS

Why Windows Containers?

Windows runs the majority of workloads



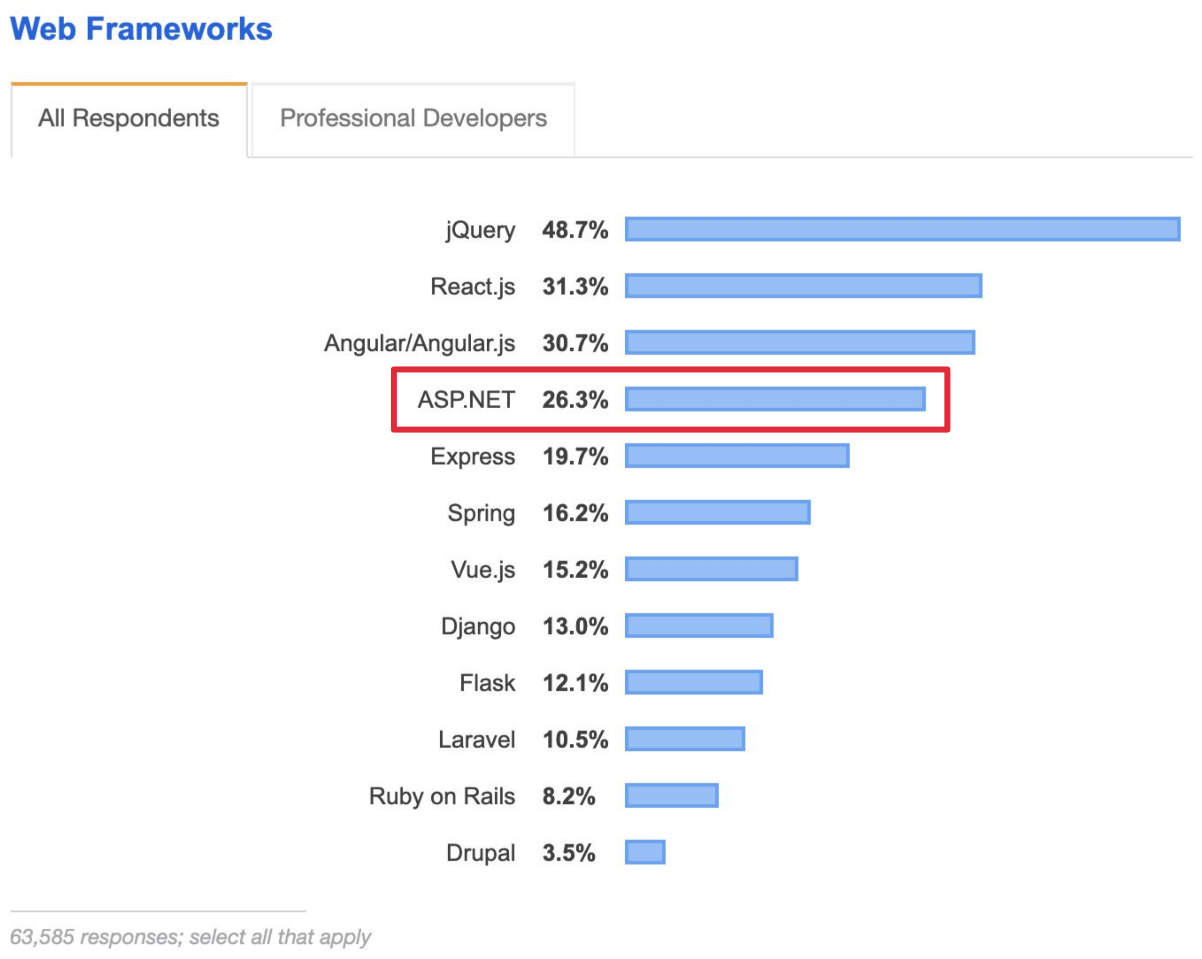


Source 1: IDC, Worldwide Operating Systems and Subsystems Market Shares, 2017: A Market Rebounds and Is Poised for Strong Gains, Doc #US43753318, May 2018

Source 2: Stack Overflow Developer Survey, 2015, 2016, 2017, 2018, 2019

MIRANTIS

.NET continues to be a top framework choice

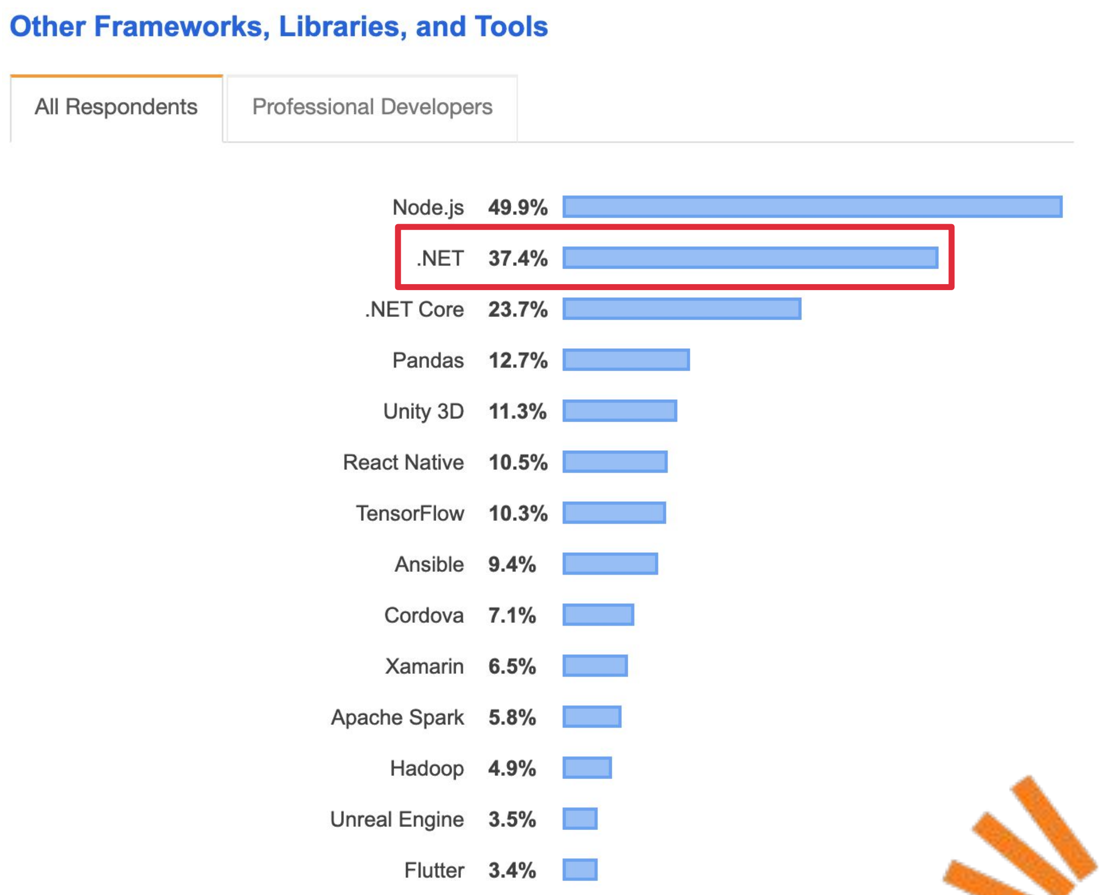


Web Frameworks

All Respondents | Professional Developers

jQuery	48.7%	
React.js	31.3%	
Angular/Angular.js	30.7%	
ASP.NET	26.3%	
Express	19.7%	
Spring	16.2%	
Vue.js	15.2%	
Django	13.0%	
Flask	12.1%	
Laravel	10.5%	
Ruby on Rails	8.2%	
Drupal	3.5%	

63,585 responses; select all that apply



Other Frameworks, Libraries, and Tools

All Respondents | Professional Developers

Node.js	49.9%
.NET	37.4%
.NET Core	23.7%
Pandas	12.7%
Unity 3D	11.3%
React Native	10.5%
TensorFlow	10.3%
Ansible	9.4%
Cordova	7.1%
Xamarin	6.5%
Apache Spark	5.8%
Hadoop	4.9%
Unreal Engine	3.5%
Flutter	3.4%

Source: Stack Overflow Developer Survey, 2019

Use Cases

Benefits across numerous initiatives

1. Consistent Operations
2. Legacy .NET Workloads
3. Cost Savings
4. Cloud Migration
5. DevOps Practices

Windows Server 2008 is now end of life

January 14, 2020
has passed

End of standard support

End of security patches

End of hotfixes

Windows Server 2008 options

Refactor & Upgrade

- Requires engineering resources
- Depending on familiarity with the app and complexity of the app, can take several weeks per application
- Once upgraded, need to repeat the same process in a few years

Extended Support Contract

- Expensive: can cost as much as 75% of license cost per year
- Kicking can down the road; will need to upgrade eventually so why wait?

"Lift and shift" servers to the public cloud

- If running on vSphere on-prem, will require a full app conversion which can take several weeks
- Only delays the inevitable as app will still be running an older OS

Containerize with Kubernetes

- Upgrade to the latest Windows Server versions
- Gain cloud portability and choice of where to deploy the app
- Future-proof apps to simplify upgrades forever

Considerations

Windows Server release channels

Long-Term Servicing Channel (LTSC) – Currently *Windows Server 2019*

- New major version of Windows Server every 2-3 years
- 5 years of mainstream support + 5 years of extended support
- Stable, predictable

Semi-Annual Channel (SAC) – Currently *Windows Server, version 1909*

- New versions twice a year (Spring + Fall)
- 18 months of support
- Faster release cadence with latest features
- Most features will be rolled into next LTSC release but not guaranteed
- Requires volume licensing or a cloud provider

Base image options

Nano Server	Server Core	Windows
`mcr.microsoft.com/windows/nanoserver:1909`	**`mcr.microsoft.com/windows/servercore:1909`**	**`mcr.microsoft.com/windows:1909`**
Greenfield and cloud-native applications	Brownfield and Legacy applications	Carries most Windows OS components
.NET Core	.NET Framework	Win 32 APIs
~100 MB	~2 GB	~4 GB

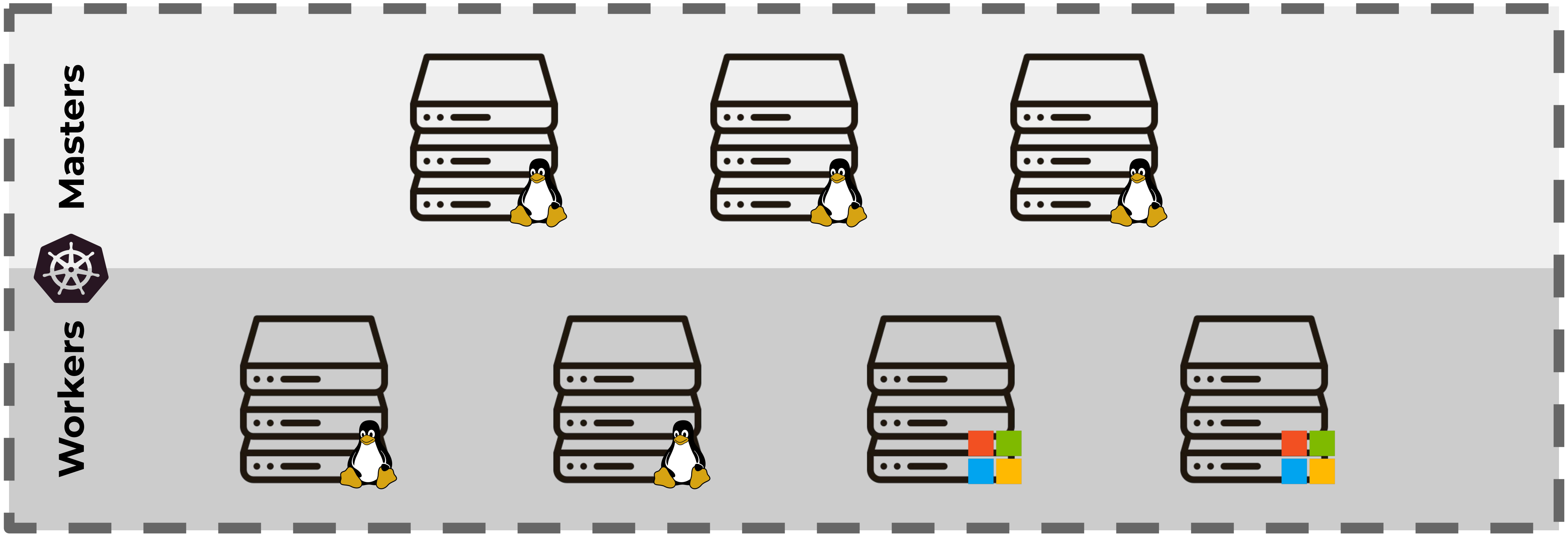
Version Compatibility

Container OS Version	Host OS Version		
	Windows Server 2019 Builds 17763.*	**Windows Server, version 1903** Builds 18362.*	**Windows Server, version 1909** Builds 18363.*
Windows Server 2019 Builds 17763.*	P H	H	H
Windows Server, version 1903 Builds 18362.*	✗	P H	H
Windows Server, version 1909 Builds 18363.*	✗	✗	P H

P **Process Isolation**

H **Hyper-V Isolation**
Alpha in K8S v1.17

Kubernetes clusters with Windows Server



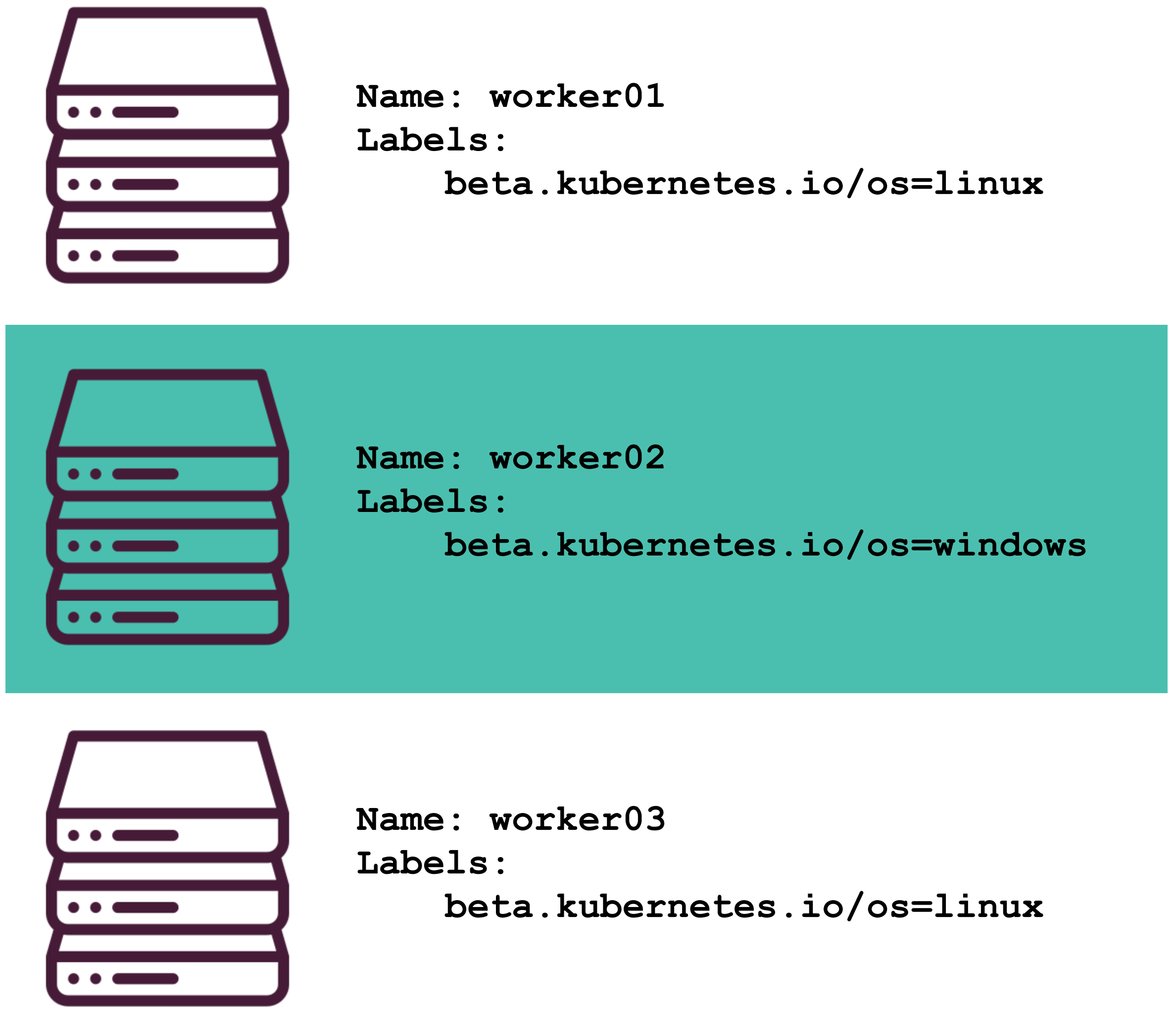
Aligning pods and nodes

1. Taint nodes to not allow Windows Pods

```
kubectl taint node \
  [NodeName] beta.kubernetes.io/os=windows:NoSchedule
```

2. Tolerate the taint in PodSpec

```
 ...
    spec:
      containers:
        - name: iis
          image: iis
      tolerations:
        - key: "os"
          operation: "equal"
          value: "windows"
          effect: "NoSchedule"
      nodeSelector:
        beta.kubernetes.io/os: windows
        node.kubernetes.io/windows-build: "10.0.17763"
```



```
Name: worker01
Labels:
    beta.kubernetes.io/os=linux
```

```
Name: worker02
Labels:
    beta.kubernetes.io/os=windows
```

```
Name: worker03
Labels:
    beta.kubernetes.io/os=linux
```

Keep in mind

Windows Server 2019

node Selector

Lack of Privileged Containers

Linux Masters

Higher Resource Needs Min 200Mi

History of Windows in Kubernetes

12/2017 **v1.9**	03/2019 **v1.14**	06/2017 **v1.15**	09/2017 **v1.16**	12/2019 **v1.17**
Beta CNI begins to take shape	**Stable** Initial support for Windows Server worker nodes	**gMSA** Alpha support for Group Managed Service Accounts (gMSA)	**Improvements** Beta support for gMSA, Alpha support for CSI	**RunAs** Beta support for `RunAsUserName`

Identity considerations

How do users authenticate to the application?

- Basic Authentication
- Forms Authentication
- Integrate Windows Authentication

How does the application authenticate to resources?

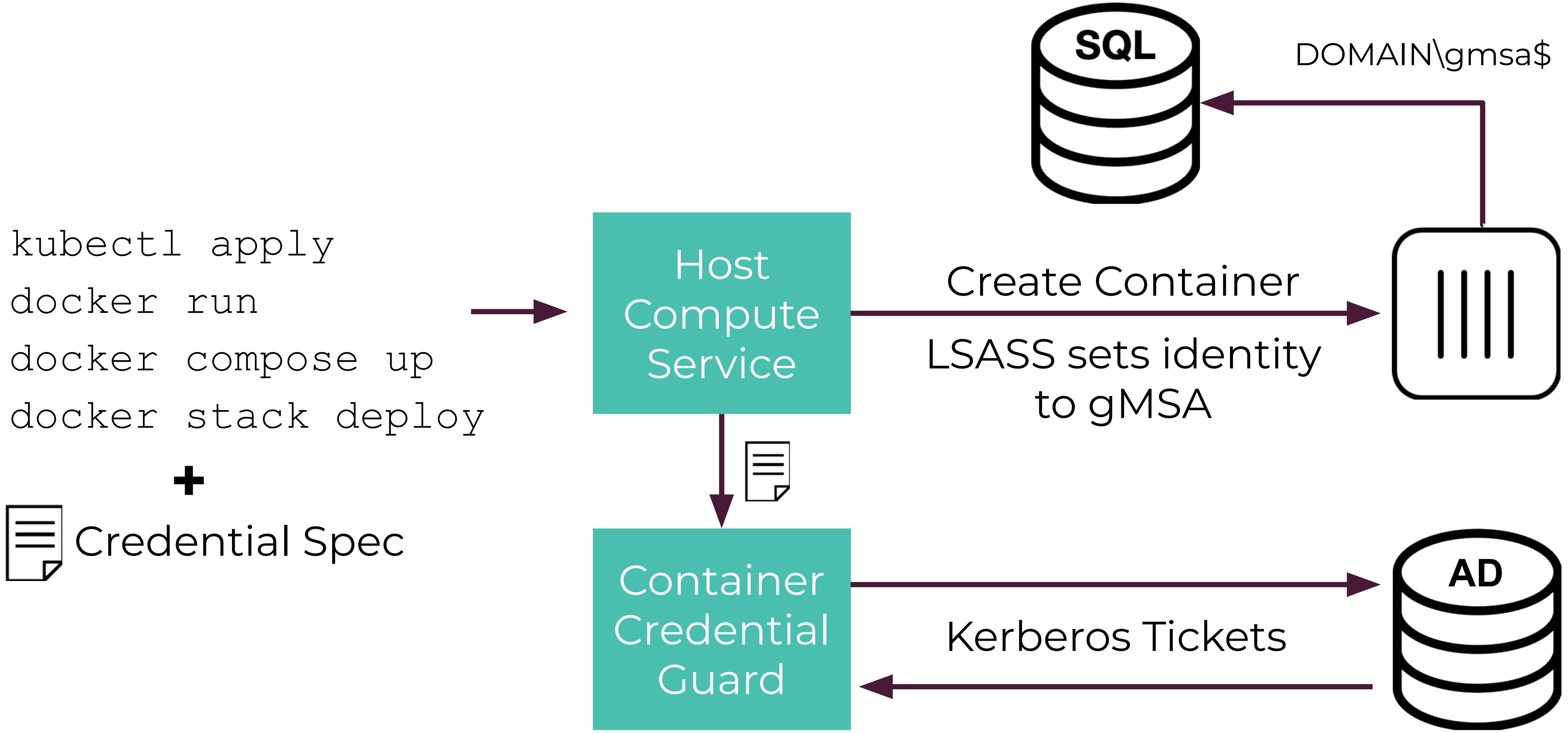
- Can the pod resolve a resource address?
- Is a Group Manage Service Account (gMSA) needed?
- Do worker nodes need to be domain joined?

Using AD with Windows Containers

```
kubectl apply
docker run
docker compose up
docker stack deploy
```

+

Credential Spec



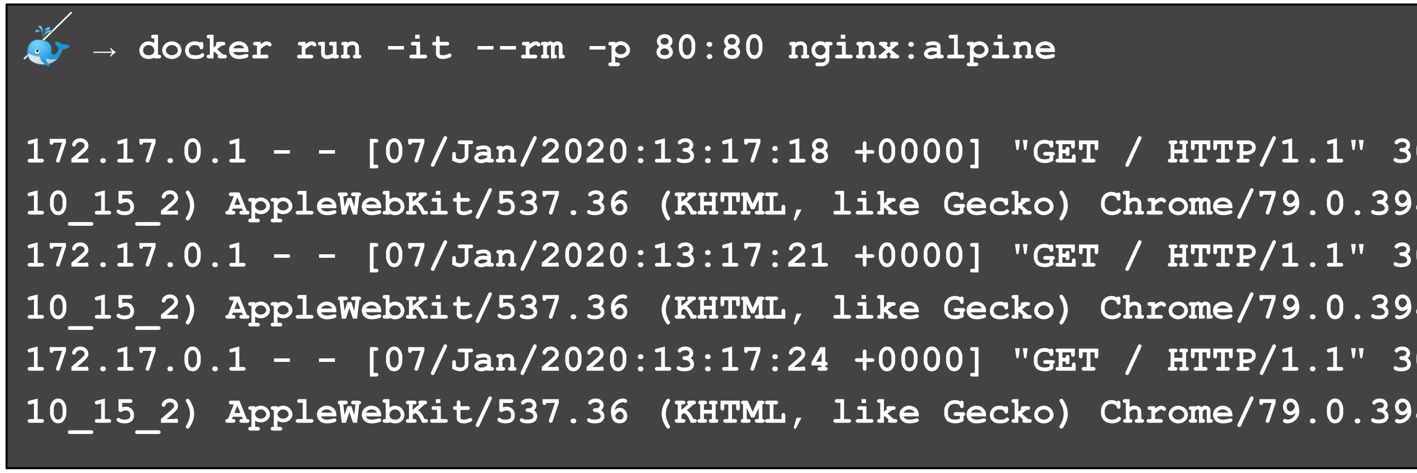
Sample Credential Spec YAML

```yaml
apiVersion: windows.k8s.io/v1alpha1
kind: GMSACredentialSpec
metadata:
 name: gmsa-webapp-1 # used for reference
credspec:
 ActiveDirectoryConfig:
 GroupManagedServiceAccounts:
   - Name: WebApp1 # GMSA account Username
 Scope: CONTOSO # NETBIOS Domain Name
 CmsPlugins:
   - ActiveDirectory
 DomainJoinConfig:
 DnsName: contoso.com # DNS Domain Name
 DnsTreeName: contoso.com # DNS Domain Name Root
 Guid: 244818ae-87ac-4fcd-92ec-e79e5252348a # GUID
 MachineAccountName: WebApp1 # GMSA account Username
 NetBiosName: CONTOSO # NETBIOS Domain Name
 Sid: S-1-5-21-2126449477-2524075714-3094792973 # GMSA SID
```

Logging considerations

Linux Applications

Log to STDOUT

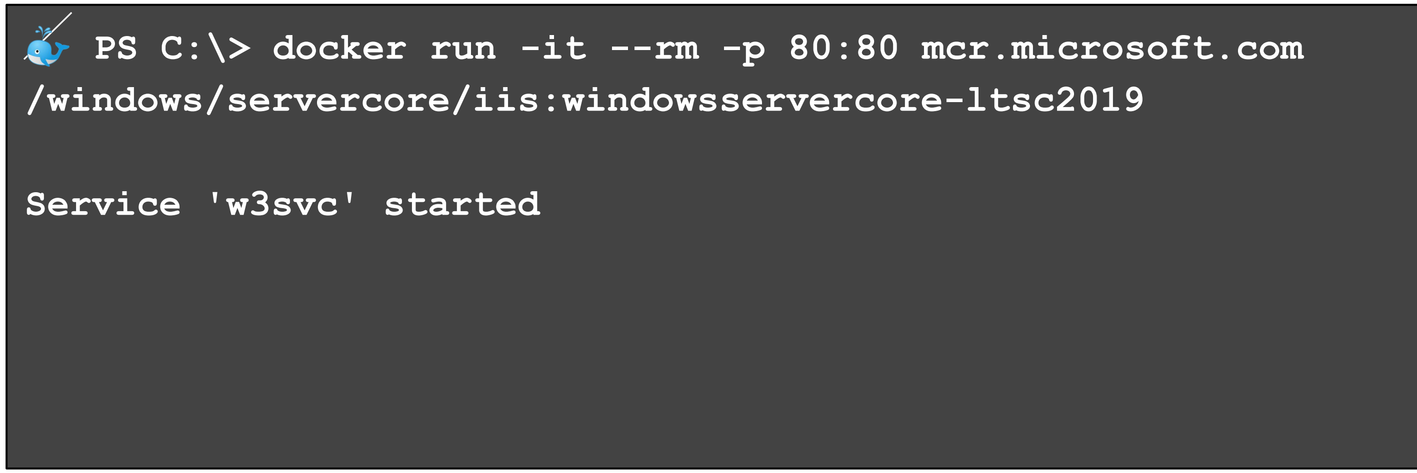


```
→ docker run -it --rm -p 80:80 nginx:alpine

172.17.0.1 - - [07/Jan/2020:13:17:18 +0000] "GET / HTTP/1.1" 3
10_15_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.39
172.17.0.1 - - [07/Jan/2020:13:17:21 +0000] "GET / HTTP/1.1" 3
10_15_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.39
172.17.0.1 - - [07/Jan/2020:13:17:24 +0000] "GET / HTTP/1.1" 3
10_15_2) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/79.0.39
```

Windows Applications

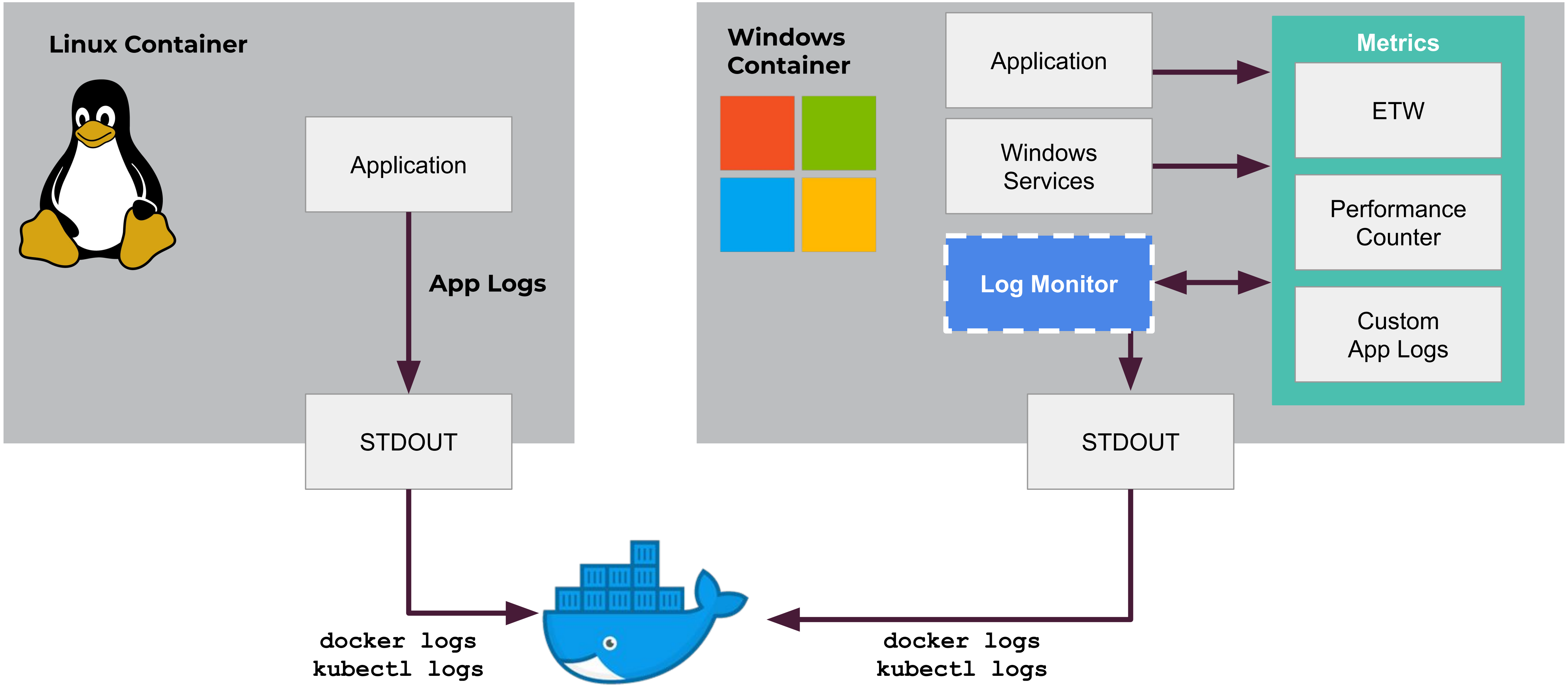
Log to ETW, Event Log, & custom files



```
PS C:\> docker run -it --rm -p 80:80 mcr.microsoft.com
/windows/servercore/iis:windowsservercore-ltsc2019

Service 'w3svc' started
```

Using the LogMonitor tool



LogMonitor tool roadmap

- Rotating log support
- Environment variable configuration support
- ConfigMap support
- Integrations with log aggregation services at scale
- Configuration updates during container runtime
- Performance
- Sidecar usage patterns
- Log driver support

https://github.com/microsoft/windows-container-tools/tree/master/LogMonitor

MIRANTIS

Demonstration

Persistent storage considerations

What data is required for your application?

- Is that data persistent?
- How large is the data?
- Databases? File Shares? Local disk locations?

Move towards databases when possible

Extract sensitive values

Identify sensitive components of applications

- Passwords
- Connection Strings
- Certificates

Utilize Kubernetes Secrets

- Clean separate between application and configuration
- RBAC-enabled to ensure proper access

State of K8S storage with Windows

- In-tree and FlexVolume plugins available today
  - File-based cloud volumes
    - Azure File through SMB
  - Block based cloud volumes
    - Azure Disk
    - GCE Persistent Disk
    - AWS EBS (WIP)
  - iSCSI Support (WIP)

- External Provisioners coming soon
- Container Storage Interface (CSI)
  - Becoming the standard for Linux containers
  - Support for Windows is coming but not ready

Summary

Summary

1

Containerize legacy applications to gain agility & cost savings

2

Start small & develop muscle around Kubernetes

3

Consider identity & storage needs early

Resources

SIG-Windows
https://github.com/kubernetes/community/tree/master/sig-windows

K8S Windows Development Kanban
https://github.com/orgs/kubernetes/projects/8

Microsoft Documentation
https://docs.microsoft.com/en-us/virtualization/windowscontainers/kubernetes/getting-started-kubernetes-windows

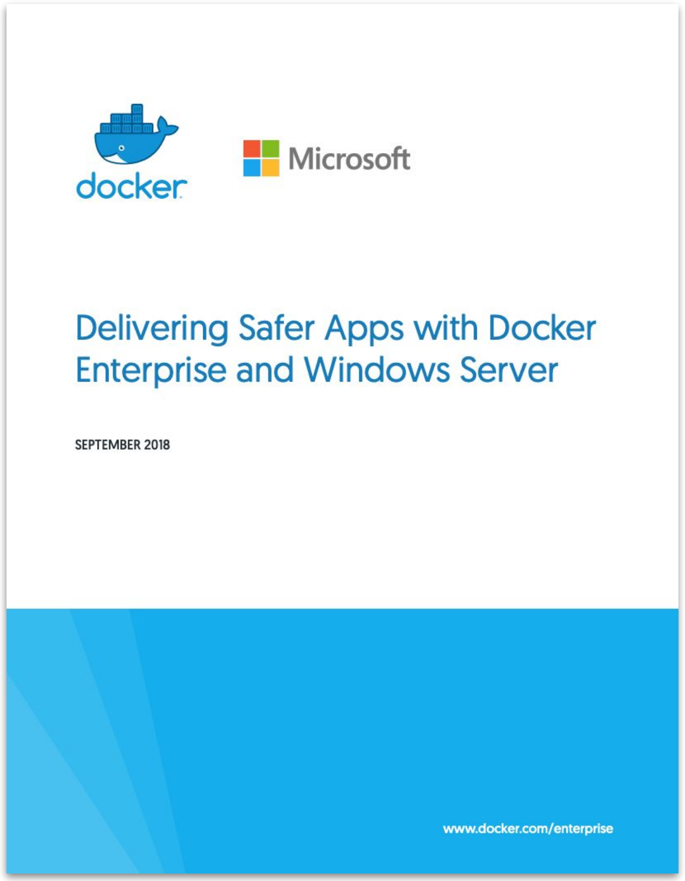
Kubernetes Documentation
https://kubernetes.io/docs/setup/production-environment/windows/intro-windows-in-kubernetes/

Resources

White Paper

Delivering Safer Apps with Docker Enterprise and Windows Server

http://bit.ly/docker-enterprise-windows-server



docker Microsoft

Delivering Safer Apps with Docker Enterprise and Windows Server

SEPTEMBER 2018

www.docker.com/enterprise

Thanks!

Contact us at:
mirantis.com/contact

KubeCon 2019 Talks

- Introduction to Windows Containers in Kubernetes - Michael Michael, VMware & Deep Debroy, Docker
- Day 2 Operations with Windows Containers - Michael Michael, VMware & Patrick Lang, Microsoft
- Superpowers for Windows Containers - Deep Debroy & Jean Rouge, Docker
- Storage Provisioning for Kubernetes on Windows - Anusha Ragunathan & Jean Rouge, Docker