Preity Gupta

containeriZe .Net 3.5 and 4.0 Applications to Azure

Contents

[Solution Description 2](#_Toc505080497)

[Solution Approach 2](#_Toc505080498)

[Use Case 1: Customer wanting to lift and shift application first to Cloud and then asking HCL to Upgrade (4.6) and Modernize. 2](#_Toc505080499)

[Prerequisites for deployment to ACS 2](#_Toc505080500)

[Process and Deployment Diagram 3](#_Toc505080501)

[Characteristics 3](#_Toc505080502)

[Challenges 4](#_Toc505080503)

[Exception 4](#_Toc505080504)

[Process of Upgrading .Net Framework 3.5 to 4. 6 or higher 4](#_Toc505080505)

[Use Case 2: Customer wanting to lift and shift application which is active (continued changes going on) and the customer is not interested to modernize. CI/CD needs to be enabled for that app. 4](#_Toc505080506)

[Prerequisites for deployment to ACS using CI/CD 4](#_Toc505080507)

[Deployment Diagram 5](#_Toc505080508)

[Characteristics 5](#_Toc505080509)

[Challenges 5](#_Toc505080510)

[Exception 5](#_Toc505080511)

[Use Case 3: Customer wanting to lift and shift application which is active and asks HCL to modernize (Upgrade to 4.6) it with minimal effort and run it on containers 6](#_Toc505080512)

[Value Added 6](#_Toc505080513)

[Use Case 1 and 2 Containerize benefits 6](#_Toc505080514)

[Use case 1 and 2 benefits of upgrading to .Net Framework 4.6 6](#_Toc505080515)

[Use Case 3 benefits of VSTS with Container Deployment 7](#_Toc505080516)

[Accelerators used and its purpose 7](#_Toc505080517)

[Activity breakup and efforts required 7](#_Toc505080518)

[Demo link 8](#_Toc505080519)

[Network Configuration for Kubernete 8](#_Toc505080520)

[Storage Configuration for Kubernete 9](#_Toc505080521)

[Summary 10](#_Toc505080522)

[Appendixes 11](#_Toc505080523)

# Solution Description

Use case 1: Customer wanting to lift and shift application first to Cloud and then asking HCL to Upgrade (4.6) and Modernize.

Use-case 2: Customer wanting to lift and shift application which is active (continued changes going on) and the customer is not interested to modernize. CI/CD needs to be enabled for that app.

Use-case 3: Customer wanting to lift and shift application which is active and asks HCL to modernize (Upgrade to 4.6) it with minimal effort and run it on containers.

He does not want to rewrite the application. Here the customer is interested to know how the containers is beneficial for .Net framework 3.5 and 4.0.

# Solution Approach

Method of Containerizing existing .NET applications with Windows Containers

|  |  |  |
| --- | --- | --- |
|  | Methods | Applicability |
| 1. | Containerize application by using Visual Studio 2017 Tools for Docker (Visual Studio 2017 or later versions) | Applicable for .Net framework 3.5 and above versions of web applications. Docker CLI method can be used but this method is more faster and easy. |
| 2. | Containerize application by manually adding a Dockerfile, and then using the Docker CLI | Applicable for .Net framework 3.0 and below versions of web applications only i.e. legacy applications. |
| 3. | Containerize application by using the Img2Docker tool (an open-source tool from Docker) | Applicable for Windows: 2-3 tier IIS and ASP.NET applications, limited external dependencies. For Linux:  2-3 tier Java apps running frameworks suited for isolation within container, LAMP stack apps, limited external dependencies  But it is applicable in case of end of life scenario where no further changes are required. |

# Use Case 1: Customer wanting to lift and shift application first to Cloud and then asking HCL to Upgrade (4.6) and Modernize.

In this scenario, Visual Studio Tools for Docker is applicable. In Visual Studio 2017, Docker support is included by default but for Visual Studio 2015, Update 3 need to be installed to enable Docker Support. For .Net Framework 3.5 and above till 4.7.1, Docker for Windows can be installed from <https://docs.docker.com/docker-for-windows/>.

## Prerequisites for deployment to ACS

* Visual Studio 2017 or VS 2015 with Update 3
* Docker for Windows
* Kubernetes cluster for Windows (Windows Server 2016)
  + To create Kubernete cluster
    - az acs create --orchestrator-type=kubernetes --resource-group myResourceGroup --name= myK8sCluster --agent-count=2 --generate-ssh-keys --windows --admin-username azureuser --admin-password myPassword12
  + To connect with Kubernete cluster on Azure from client
    - az acs kubernetes install-cli
    - az acs kubernetes get-credentials --resource-group=myResourceGroup -- name=myK8sCluster
    - Kubectl proxy
* Container Registry
* Dockerfile

FROM microsoft/ aspnet:4.7.1-windowsservercore-ltsc2016

ARG source

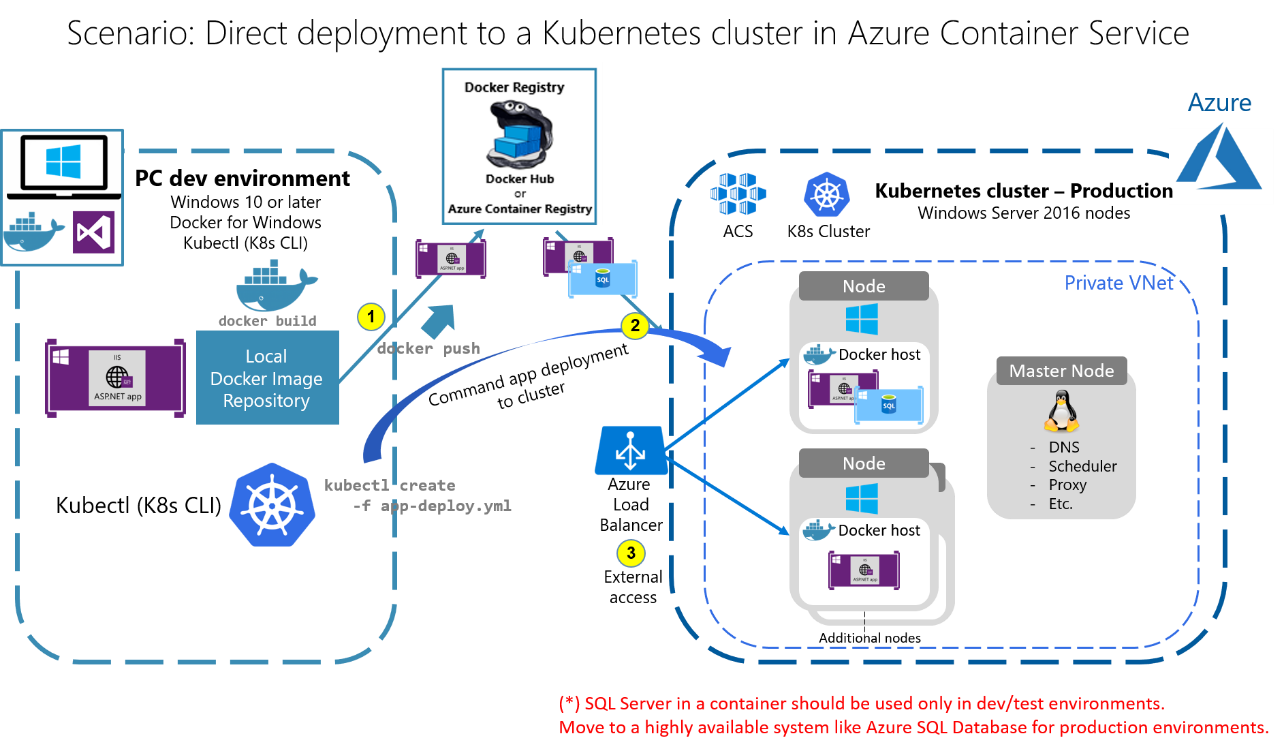
WORKDIR /inetpub/wwwroot

COPY ${source:-obj/Docker/publish}.

* Deployment file

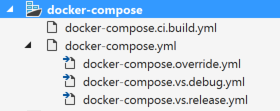
## Process and Deployment Diagram

* Migrate the LocalDB to SQL Server in Azure
* Using the Docker tools in Visual Studio 2017, add the Docker files
* Publish Docker Images to Azure Container Registry (ACR)
* Push the new Docker images from ACR to Azure Container Services -Kubernetes Cluster



## Characteristics

* Visual Studio Tools for Docker is based on the same Docker CLI, but it is easier to get started, simplifies the process, and provides greater productivity for the build, run, and compose tasks.
* It enables to execute and debug containers via simple actions like F5 and Ctrl+F5 from Visual Studio.
* In addition to being able to run and debug a single container, group of containers (a whole solution) can be ran at the same time if they are defined in the same docker-compose.yml file at the solution level.
* Once Docker support is added to the solution in Visual Studio, new node tree in Solution Explorer with the added docker-compose.yml files will appears:



* Azure Container Services supports high scalability, automated deployments and versioning facilities by using orchestrator like Kubernetes, Docker Swarm, or DC/OS and Service fabric. Scalability can be inner-scalability and global scalability. Inner-scalability in which application is scale-out based on the number of container instances required. global scalability in which number of nodes or VMs in the cluster are scaled out.
* zero code change

## Challenges

* Database associated with an application need to be migrated either to Azure SQL or deployed to ACS using deployment script depending on the structure of database.
* In case of continuous changes in code, deployment script need to be ran manually on regular basis.
* In case of discrepancy, backup of production environment for rollback of code from production is manual and time consuming. This causes downtime and loss to business.

## Exception

* Desktop application cannot be dockerized using this method
* In case application is in a form of Website instead of Web application then it need to first converted to Web application

## Process of Upgrading .Net Framework 3.5 to 4. 6 or higher

Go to Project right click on it --> Properties ---> Application -- Select Target Framework i.e 4.6 and then click Select. This will upgrade the application to .Net Framework to 4.6.

# Use Case 2: Customer wanting to lift and shift application which is active (continued changes going on) and the customer is not interested to modernize. CI/CD needs to be enabled for that app.

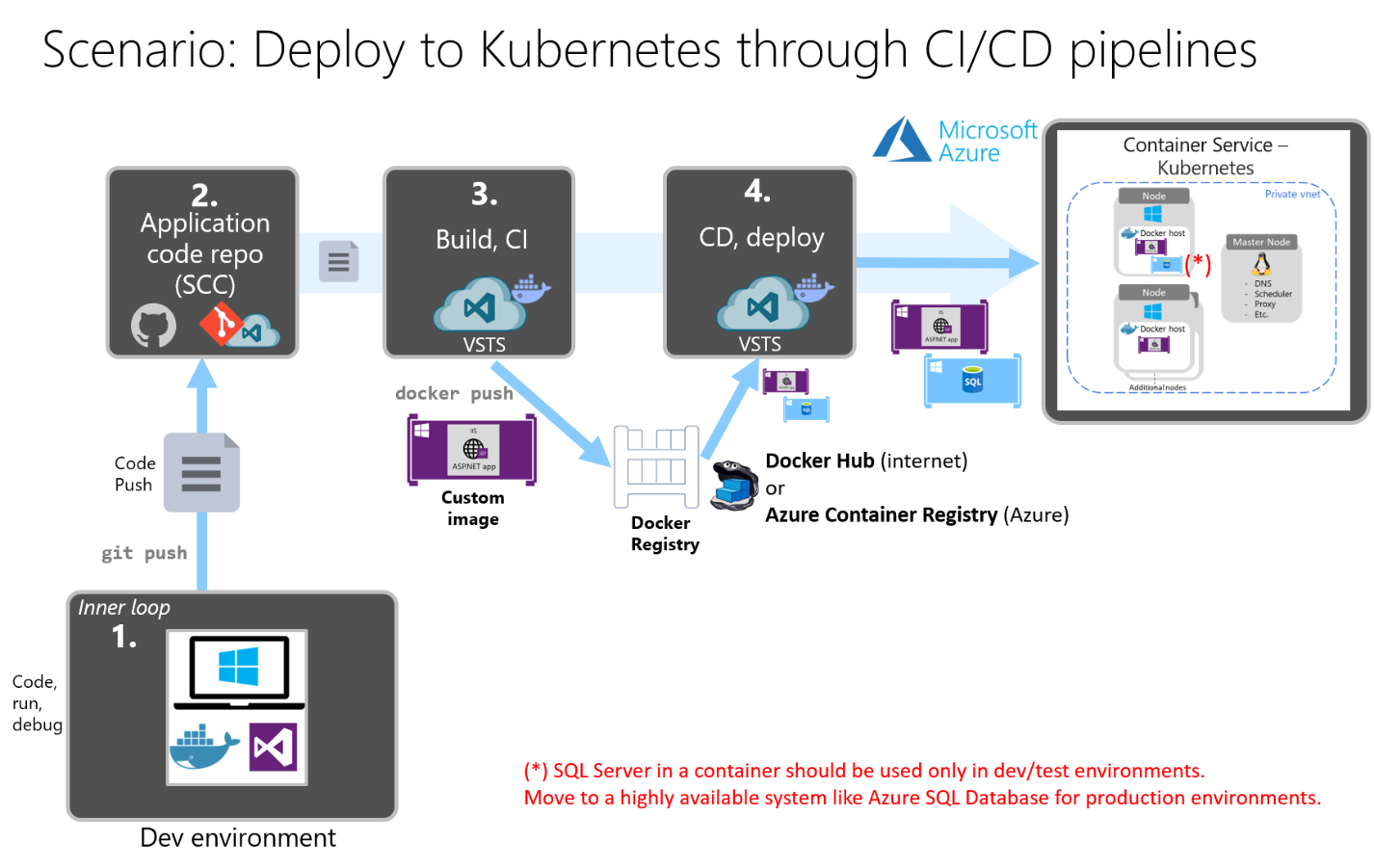
Under this scenario, .Net framework 3.5 and 4.0 can easily be deployed to Azure Container Services(ACS) without any upgradations using same process as explained in Use Case 1.

In case CI/CD is need then using Visual Studio 2017, continuous integration and continuous deployment(CI/CD) can be implemented using Visual Studio Team foundation Server(VSTS).

## Prerequisites for deployment to ACS using CI/CD

* All the perquisites mentioned in Use Case 1
* VSTS account login and admin rights.

## Deployment Diagram



## Characteristics

* VSTS allows to centralised the repository of code at one location in secure manner.
* VSTS not only automates the build, testing and deployment of your app, it gives complete traceability to see everything in the build including changes to code, reviews, and test results. It delivers more value to users with more frequent deployments of a higher quality.
* VSTS allows deployment from anywhere VSTS (Git or TFVC), GitHub, Bitbucket, private repo, or Subversion.
* VSTS allows deployment to any platform such as Azure, AWS, GCP, app stores such as Microsoft, Android, iOS, or Windows and Linux systems.
* VSTS supports build from many languages like Java, .NET, PHP, Python, Ruby, C++ etc.
* VSTS supports deploy to websites, Service Fabric, Docker container orchestrators, virtual machines, and more.

## Challenges

* Build time becomes longer with the addition of more components to applications, the more lines of code, and the more tests and routines run as part of your build process.

## Exception

* Desktop application cannot be containerized using this method.
* In case application is in a form of Website instead of Web application then it needs to be converted to Web application first.

# Use Case 3: Customer wanting to lift and shift application which is active and asks HCL to modernize (Upgrade to 4.6) it with minimal effort and run it on containers

Under this scenario, Upgradation is done before lift and shift process using Visual Studio Tools for Docker. It is already stated in use case 1 that upgradation to .Net framework 4.6 does not affect deployment to Docker. From application point of view, upgradation to .Net framework 4.6 helps in better allocation and release process of memory, improvement in performance due to parallel computing, enhanced security and eases the efforts of developers in development and support. This use case also has same characteristics, challenges, exceptions and benefits as user case 1.

# Value Added

## Use Case 1 and 2 Containerize benefits

* **Consistent:** Containers include the application and all its dependencies. The application executes the same code, regardless of computer, environment or cloud.
* **Lightweight:** Containers start quickly and use a minimal amount of RAM by using a minimal abstraction over the host operating system and sharing common resources across containers.
* **Sharing:** Container images are easy to share via Docker Hub, the Docker Store, and private Docker registries, such as the Azure Container Registry.
* **Simple yet powerful:** The DockerFile format (the recipe for container images) is a simple format that enables powerful scenarios: neatly marries operating-system and container-specific commands and surfaces the creation of Docker image layer.
* **Infrastructure Cost**: Container helps reducing 50% cost by server consolidation, reduction in virtualized instances, and lower overhead costs.
* **Operational Efficiencies**: Containers reduces operational cost by improving the way applications are deployed and maintained. Built in capabilities improve reliability and availability.
* **Application Deployment Rate:** Containers allows organizations to ship 13x more frequently as the path from developer’s laptops to production is simplified and the development pipeline become faster using CI/CD.
* **Portability:** Containers are modular and portable. Docker containers are supported on any server operating system (Linux and Windows), in any major public cloud (Microsoft Azure, Amazon AWS, Google, IBM), and in on-premises and private or hybrid cloud environments.

## Use case 1 and 2 benefits of upgrading to .Net Framework 4.6

From Docker point of view, there is no benefit of upgradation but from application point of view, there are following benefits:

* **Better memory resources** – the latest upgrade has improved the speed in which an application allocates and releases memory.
* **Parallel computing** – the latest version introduces a new programming tool called parallel computing which gives the application a great deal more capabilities to perform operations simultaneously, thus increasing performance
* **Improved security - .**NET 4.6 provides improved security andintroduces new security methods which simplify the process for the developer
* **Easier to develop.** The updates and additional features in .NET 4.6 simplifies the process for developers to code.

## Use Case 3 benefits of VSTS with Container Deployment

* Whole process is automated thus reduces manual effort and manual intervention.
* Time between build and deployment is reduced thus ensure faster time-to-delivery that improves ROI.
* Improve collaboration between teams (Business / Dev / Ops) by improving the transparency required for effective decision making.
* Provides stable/reliable operating environments.
* Enables early detection and faster correction of defects that helps provide the best services and robust features to customers
* Enables continuous Release and Deployment, Continuous Testing, and Continuous Monitoring to continuously deliver quality software, reduced go-to-market timelines, and adaptation of shorter release cycles.
* Zero code change and zero downtime.

# Accelerators used and its purpose

|  |  |
| --- | --- |
| Tools Used | Purpose |
| Docker for Windows | It enables Docker support and debugging and running of application. |
| Visual Studio 2017 or VS 2015 with Update 3 | * It helps debug and run application on local Docker * It helps pushing image to Azure Container Registry |
| Kubernetes cluster for Windows (Windows Server 2016) | * Kubernete enables deployment of multiple applications in same cluster * Kubernete allows communication between different applications * Kubernete enables scaling individually * Kubernete provides complete management and automatic load balancing facilities |
| Container Registry | It is used as repository to store images of applications |
| Dockerfile | It helps creating image of application |
| Deployment file | It helps pulling image from Azure Container Registry to Azure Container Services |
| VSTS (for Use Case 3) | It helps automate the process of pushing image to Azure Container Registry and pulling it back to Azure Container Services |

# Activity breakup and efforts required

|  |  |
| --- | --- |
| Activities | Efforts |
| Installation of Docker of Windows Software | Go to <https://docs.docker.com/docker-for-windows/> URL and download stable version and install in client system. Within 5 minutes’ system is compliant for Docker support. |
| Setting up of Kubernete cluster on Azure | Using creation command from command prompt, within 15 minutes kubernete Azure Container Services is active with all configurations |
| Interacting with Kubernete from client system | Using command from command prompt for installing command line interface, setting up credentials on client system takes 5 minutes |
| Setting up Container Registry | Going to Azure portal and setting up Azure Container Registry takes only 5 minutes to configure it. |
| Creation of Dockerfile | Dockerfile is created automatically |
| Creation of Deployment file | Deployment file can be written in 5 to 10 minutes |
| Setting up of VSTS Account | Creation of admin account takes 5 minutes |
| Configuring application for VSTS | To set up VSTS as repository takes approx. 10 minutes |
| Creation of Build and release statement into VSTS for deployment to Kubernete | To set up build and release statement take 10 minutes |

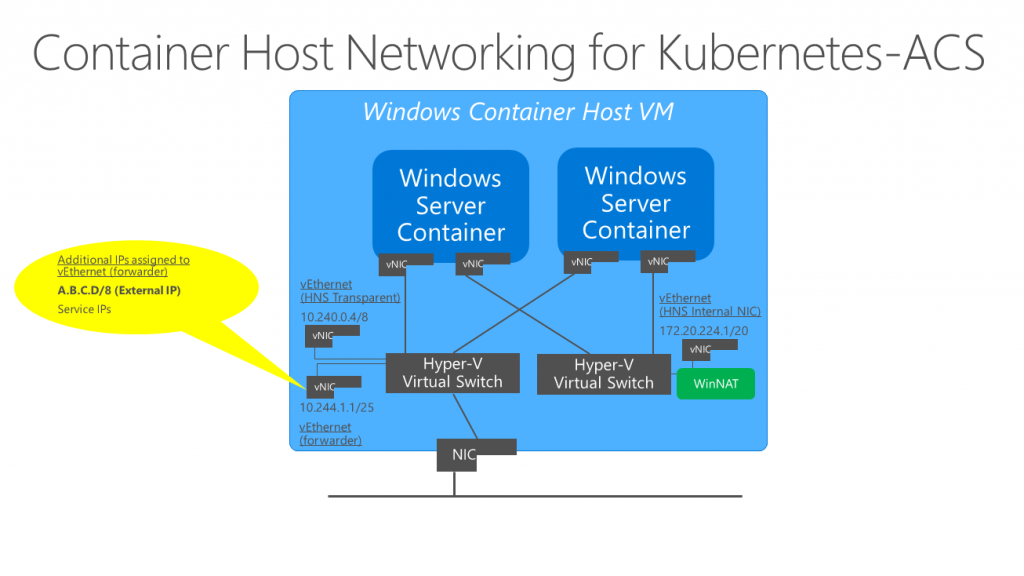
# Demo link

<https://kx.hcl.com/cop/MWSOA%20Community/BusinessDevelopment/COE/SitePages/Home.aspx?RootFolder=%2Fcop%2FMWSOA%20Community%2FBusinessDevelopment%2FCOE%2FShared%20Documents%2FNewApps%2FApp%2DMod%2DCloud%2FModernization%2FPOCs&FolderCTID=0x01200028A57BAF38A61244AD849F061EC7FF1B&View=%7B5619EBD5%2D7325%2D4EAD%2D9F05%2DDFEADB0C64C6%7D>

Filename: Containerize 3.5 and 4.0 application to ACS.zip

# Network Configuration for Kubernete

Using HNS, one transparent and one NAT network is created on each Windows container host for inter-Pod and external communication respectively. Two container endpoints – connected to the Service and Pod networks – are required for each Windows container which will participate in the Kubernetes service. Static routes must be added to the running Windows containers themselves on the container endpoint attached to the service network.

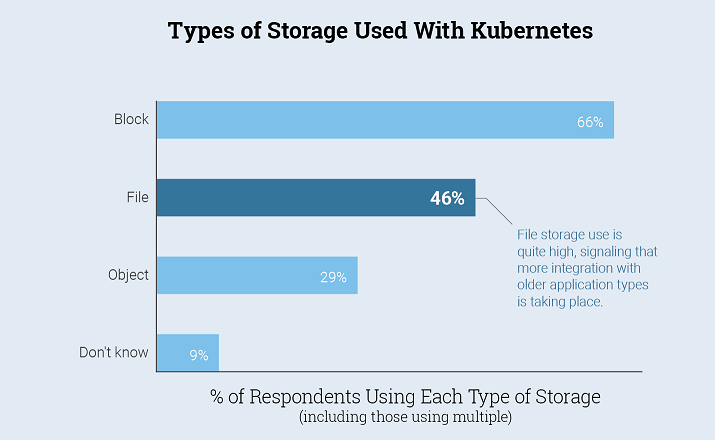
[](https://msdnshared.blob.core.windows.net/media/2017/04/Windows-Container-Host-ACS-Kubernetes.png)

In the absence of ACS-managed User-Defined Routes, Out-of-Band (OOB) configuration of these routes need to be realized in the Cloud Service Provider network, implemented using the “routing” interface of the Kubernetes cloud provider, or connected via overlay networks.

Other solutions include using the HNS overlay network driver for inter-Pod communication or using the OVS Hyper-V switch extension with OVN Controller.

# Storage Configuration for Kubernete

Block storage occupies two-thirds (66 percent) for Kubernetes implementations. Few deployments are downgraded to only one type of logical storage, so (46 percent) cited by file storage. Newer cloud-native applications with microservices architectures that utilize databases or data structures don’t need a file system because they are not interacting with data through an operating system hence 46 percent figure is quite high but it indicates that it has more integration with older application. Object storage is used by 29 percent of respondents, which is relatively high compared since object storage is scalable, developers working on distributed systems use it. In addition, object storage is often used to deliver static content for websites.



# Summary

Container-based deployments have quickly become the preferred approach for managing the build and release of complex applications. Container-based development is both productive and compelling, and it reduces the number of moving parts, which historically was the cause of many mistakes and system challenges. Containers provide exciting new capabilities for application deployment at scale, which can deliver highly reliable and secure systems.

# Appendixes

<https://docs.microsoft.com/en-us/dotnet/standard/modernize-with-azure-and-containers/lift-and-shift-existing-apps-devops/reasons-to-lift-and-shift-existing-net-apps-to-cloud-devops-ready-applications>

<https://docs.microsoft.com/en-us/dotnet/standard/modernize-with-azure-and-containers/walkthroughs-technical-get-started-overview>

<https://docs.microsoft.com/en-us/dotnet/standard/modernize-with-azure-and-containers/walkthroughs-technical-get-started-overview>