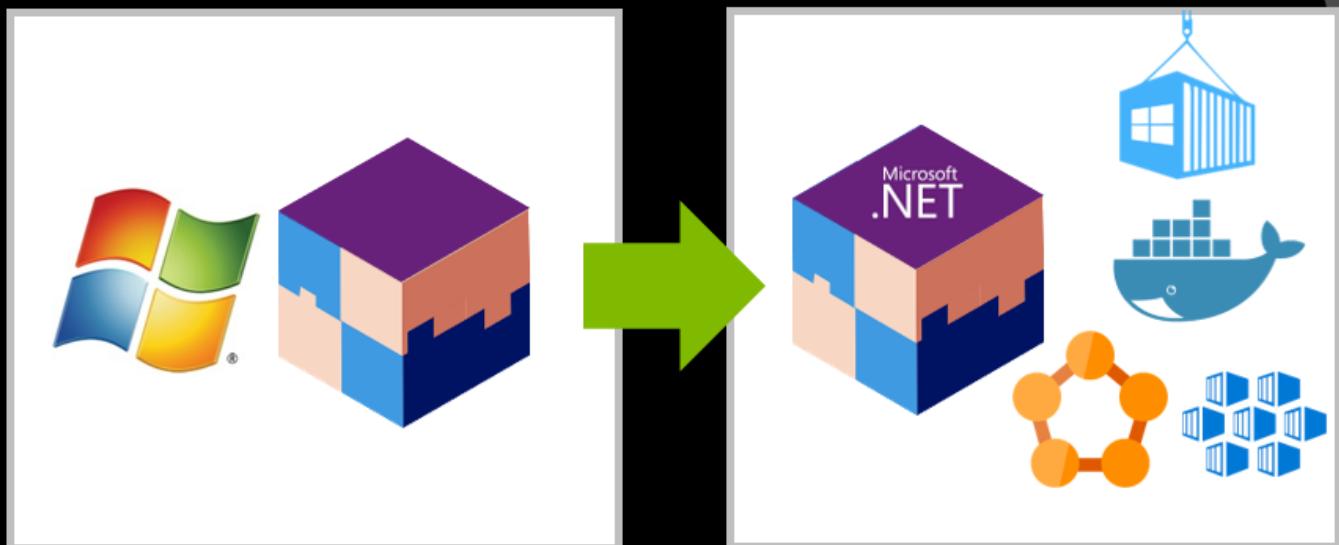


Modernizar aplicaciones .NET existentes con la nube de Azure y Contenedores Windows



Cesar de la Torre
Microsoft Corp.

Modernizar aplicaciones .NET existentes con la nube de Azure y Contenedores Windows (v1.0)

PUBLICADO POR

Microsoft Press and Microsoft DevDiv
Divisions of Microsoft Corporation
One Microsoft Way
Redmond, Washington 98052-6399

Copyright © 2017 by Microsoft Corporation

Todos los derechos reservados. Ninguna parte del contenido de este libro puede reproducirse de ninguna forma ni por ningún medio sin la autorización por escrito del editor.

Este libro está disponible gratuitamente en forma de libro electrónico (eBook) a través de múltiples canales de Microsoft como <http://dot.net/architecture>

Si tiene alguna pregunta relacionada con este libro, envíe un email a dotnet-architecture-ebooks-feedback@service.microsoft.com

Este libro se proporciona "tal cual" y expresa las opiniones y opiniones del autor. Los puntos de vista, opiniones e informaciones expresadas en este libro, incluidas las direcciones URL y otras referencias a sitios web de Internet, pueden cambiar sin previo aviso.

Algunos ejemplos ilustrados aquí se proporcionan solo a título ilustrativo y son ficticios. No se pretende ni debe inferirse ninguna asociación o conexión real.

Microsoft y las marcas que figuran en <http://www.microsoft.com> en la página web "Trademarks" son marcas comerciales del grupo de compañías de Microsoft. Todas las demás marcas son propiedad de sus respectivos dueños.

Autor:

Cesar de la Torre, Sr. PM, .NET Product Team, Microsoft Corp.

Participantes y editores:

Scott Hunter, Partner Director PM, .NET team, Microsoft

Paul Yuknewicz, Principal PM Manager, Visual Studio Tools team, Microsoft

Lisa Guthrie, Sr. PM, Visual Studio Tools team, Microsoft

Ankit Asthana, Principal PM Manager, .NET team, Microsoft

Unai Zorrilla, Developer Lead, Plain Concepts

Javier Valero, Chief Operating Officer at Grupo Solutio

Contenidos

Introducción	1
Acerca de esta guía	1
Camino a la nube para aplicaciones .NET existentes	1
Tecnologías clave y arquitecturas por nivel de madurez.....	5
Escenarios de migración directa	8
Lo que esta guía no cubre	10
Quién debe usar esta guía	11
Como usar esta guía	12
Aplicaciones ejemplo para modernizar aplicaciones legacy: eShopModernizing.....	12
¡Envíanos tus comentarios!.....	12
Migrar de manera directa aplicaciones .NET existentes a Azure IaaS (Preparadas para Infraestructura en la Nube)	13
Por qué migrar aplicaciones web .NET existentes a Azure IaaS	14
Cuando migrar a IaaS en lugar de a PaaS	14
Uso de Azure Migrate para analizar y migrar aplicaciones existentes a Azure	15
Usar Azure Site Recovery para migrar las VMs existentes a Azure VMs.....	16
Migrar bases de datos relacionales a Azure	18
Cuando migrar a Azure SQL Database Managed Instance.....	19
Cuándo migrar a Azure SQL Database	20
Migrar de manera directa aplicaciones .NET existentes a aplicaciones Preparadas para DevOps en la Nube.....	23
Principios y procedimientos de aplicaciones Preparadas para DevOps en la Nube.....	24
Beneficios de una aplicación Preparada para DevOps en la Nube	26
Tecnologías Microsoft en aplicaciones Preparadas para DevOps en la Nube	27
Las aplicaciones monolíticas pueden ser aplicaciones Preparadas para DevOps en la Nube.....	28
Aplicaciones nativas con aplicaciones Optimizadas para la Nube	31
¿Qué hay sobre los microservicios?	31
Cuando utilizar Azure App Service para modernizar aplicaciones .NET existentes.....	32
Como desplegar aplicaciones existentes .NET a Azure App Service	35

Validar sitios y migrar a App Service con Azure App Service Migration Assistant.....	35
Despliegue de aplicaciones .NET existentes como Contenedores Windows.....	37
¿Que son los contenedores? (Linux o Windows)	37
Beneficios de los contenedores (Docker Engine en Linux o Windows)	37
¿Qué es Docker?	38
Beneficios de los Contenedores Windows para aplicaciones .NET existentes	39
Elegir un Sistema Operativo al cual orientarse con contenedores basados en .NET	40
Tipos de Contenedores Windows.....	41
Cuando no desplegar a Contenedores Windows.....	42
Cuándo desplegar Contenedores Windows en infraestructura on-premises IaaS basada en VMs....	43
Cuándo desplegar Contenedores Windows a Azure VMs (nube IaaS).....	43
Cuando desplegar Contenedores Windows a Service Fabric	44
Cuándo desplegar Contenedores Windows a Azure Container Service (p.e., Kubernetes)	45
Construir servicios resilientes preparados para la nube: Asumir fallos transitorios en la nube	46
Tratando los fallos parciales	46
Modernizar aplicaciones con monitorización y telemetría.....	48
Monitorización de aplicaciones con Application Insights	48
Monitorizar infraestructura Docker con Log Analytics y su solución de monitorización de Contenedor	49
Modernizar el ciclo de vida de las aplicaciones, con procesos CI/CD y herramientas DevOps en la nube	51
Migrar a escenarios híbridos en la nube	53
Azure Stack	53
Tutoriales y visión técnica inicial.....	56
Lista de tutoriales paso-a-paso técnicos	56
Tutorial 1: Tour de aplicaciones legacy en eShop.....	57
Disponibilidad del tutorial técnico paso-a-paso.....	57
Visión general	57
Objetivos.....	57
Escenario	57
Beneficios	58
Siguientes pasos	58
Tutorial 2: Contenerizar aplicaciones .NET existentes con Contenedores Windows	59

Disponibilidad del tutorial técnico paso-a-paso.....	59
Visión general	59
Objetivos.....	59
Escenario.....	59
Beneficios	60
Siguientes pasos	60
Tutorial 3: Desplegar una aplicación basada en Contenedores Windows a Azure VMs.....	61
Disponibilidad del tutorial técnico paso-a-paso.....	61
Visión General.....	61
Objetivos.....	61
Escenarios.....	61
Azure VMs para Contenedores Windows	63
Beneficios	63
Siguientes pasos	63
Tutorial 4: Desplegar aplicaciones basadas en Contenedores Windows a Kubernetes en Azure Container Service	64
Disponibilidad del tutorial técnico paso-a-paso.....	64
Visión general	64
Objetivos.....	64
Escenarios.....	65
Beneficios	66
Siguientes pasos	66
Tutorial 5: Desplegar aplicaciones basadas en Contenedores Windows a Azure Service Fabric.....	67
Disponibilidad del tutorial técnico paso-a-paso.....	67
Visión general	67
Objetivos.....	67
Escenarios.....	68
Beneficios	69
Siguientes pasos	69
Conclusiones.....	70
Puntos clave	70

Introducción

Cuando se deciden modernizar aplicaciones web y moverlas a la nube, no necesariamente se tiene que redefinir la arquitectura de las aplicaciones. Redefinir la arquitectura de una aplicación utilizando un enfoque avanzado como microservicios no siempre es una opción, debido a las restricciones de coste y tiempo. Dependiendo del tipo de aplicación, redefinir la arquitectura podría no ser necesario. Para optimizar el coste-efectividad de la estrategia de migración a la nube de una organización, es importante tener en cuenta las necesidades del negocio y los requerimientos de las aplicaciones. Para ello se tendrá que determinar:

- Qué aplicaciones requieren una transformación o redefinición de arquitectura.
- Qué aplicaciones necesitan ser modernizadas parcialmente.
- Qué aplicaciones se pueden migrar directamente a la nube.

Acerca de esta guía

Esta guía se centra principalmente en escenarios de migración directa a la nube, y en la modernización inicial de aplicaciones Microsoft .NET Framework web existentes u orientadas a servicios. Migrar directamente a la nube es la acción de mover una carga de trabajo a un entorno más nuevo o más moderno, sin alterar el código de la aplicación y su arquitectura básica.

Esta guía describe cómo mover aplicaciones de servidor existentes, basadas en .NET Framework, directamente a la nube mediante la modernización de áreas específicas, sin necesidad de redefinir su arquitectura o de recodificarlas por completo.

Esta guía también destaca los beneficios de mover aplicaciones a la nube y parcialmente modernizarlas utilizando un conjunto específico de nuevas tecnologías y enfoques, como Contenedores Windows y orquestadores en Azure.

Camino a la nube para aplicaciones .NET existentes

Las organizaciones suelen elegir pasar a la nube por la agilidad y velocidad que se pueden conseguir en sus aplicaciones. Se pueden levantar hasta miles de servidores (VMs) en la nube en minutos, en comparación con las semanas que por lo general requiere configurar los servidores locales.

No existe una sola estrategia transversal para la migración de aplicaciones a la nube. La estrategia ideal dependerá de las necesidades y prioridades de cada organización, y los tipos de aplicaciones que se están migrando. No todas las aplicaciones justifican la inversión de una migración a un modelo de Plataforma como Servicio ([PaaS](#)) o el desarrollar un modelo de aplicación de [Nube Nativa](#). En muchos casos, se puede elegir una aproximación de invertir por fases o de manera incremental para el movimiento de activos a la nube, basado en cada necesidad de negocio.

Para aplicaciones modernas buscando una mayor agilidad, persistente a largo plazo, y un mayor valor para la empresa, es posible beneficiarse invirtiendo en arquitecturas nativas u optimizadas para la nube. Sin embargo, para aplicaciones existentes o activos tipo *legacy* (heredadas o antiguas), la clave es gastar el mínimo tiempo y dinero (sin redefinir arquitecturas ni llevar a cabo cambios en el código) mientras se mueven a la nube, para lograr beneficios significativos.

La Figura 1-1 muestra los posibles caminos que se pueden tomar para mover aplicaciones .NET existentes a la nube en fases incrementales.

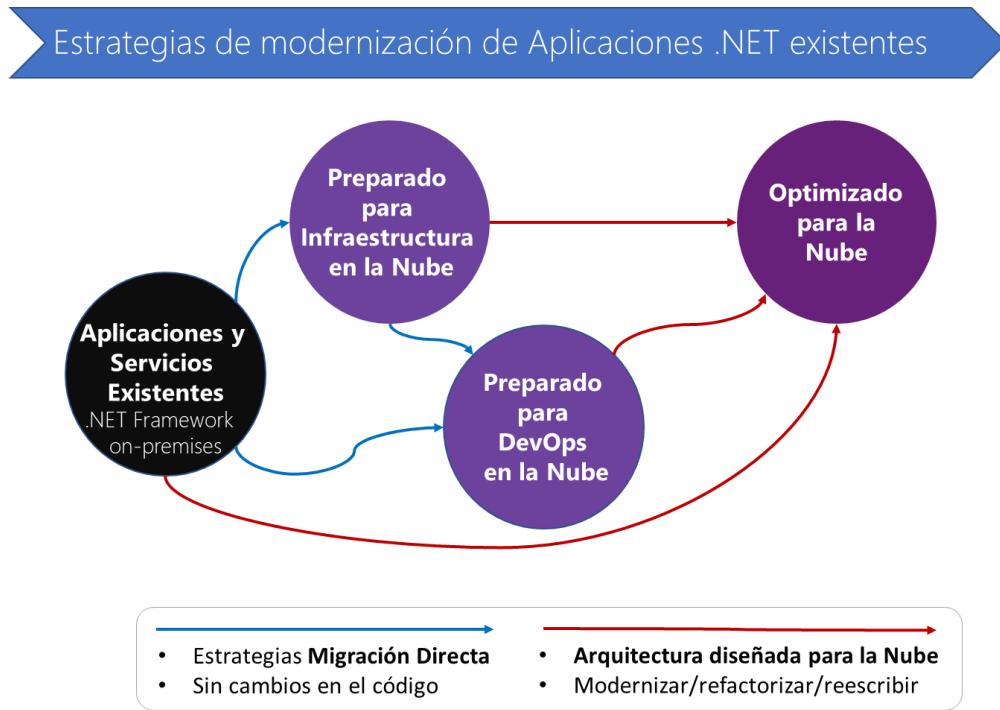


Figura 1-1. Camino para modernizar servicios y aplicaciones .NET existentes

Cada enfoque de migración tiene diferentes beneficios y justificaciones para su uso. Se puede elegir un solo enfoque cuando se migran aplicaciones a la nube, o elegir determinados componentes de múltiples enfoques. Las aplicaciones individuales no están limitadas al uso de un solo enfoque o estado de madurez. Por ejemplo, un enfoque híbrido común tendría ciertos componentes *on-premises* (en las propias instalaciones locales), además de otros en la nube.

En los dos primeros niveles de migración, se puede simplemente hacer una migración directa de la aplicación:

Nivel 1: Preparado para Infraestructura en la Nube: En este enfoque de migración, sólo se tienen que reubicar o mover las aplicaciones actuales *on-premises* a una plataforma de infraestructura como un servicio ([IaaS](#)). Las aplicaciones tienen casi la misma composición que antes, pero ahora se despliegan en máquinas virtuales en la nube.

Nivel 2: Preparado para DevOps en la Nube: En este nivel, después de una migración directa inicial, y todavía sin modificar la arquitectura o alterar el código, se pueden obtener incluso más beneficios al ejecutar la aplicación en la nube. Se mejora la agilidad en el

lanzamiento de las aplicaciones al mercado mediante el refinado de los procesos de operaciones empresariales en el desarrollo (DevOps). Esto se consigue mediante el uso de tecnologías como Contenedores Windows, que se basa en Docker Engine. Los contenedores eliminan la fricción causada por las dependencias de una aplicación cuando se despliega en múltiples etapas. Adicionalmente, los contenedores también utilizan servicios gestionados en la nube relacionados con los datos, monitorización y procesos de integración continua/despliegue continuo (CI/CD).

El tercer nivel de madurez es la última meta en la nube, aunque opcional para muchas aplicaciones y no el foco principal de esta guía:

Nivel 3: Optimizado para la Nube: Este método de migración es típicamente impulsado por necesidades de la empresa, y se enfoca a modernizar aplicaciones de misión crítica. En este nivel, se utilizan servicios PaaS para mover las aplicaciones a plataformas PaaS. Se implementan aplicaciones de tipo [Nube Nativa](#) y arquitectura de microservicios para evolucionar aplicaciones manteniendo su agilidad a largo plazo, y escalar a nuevos límites. Este tipo de modernización por lo general requiere redefinir arquitecturas específicamente para la nube. Por lo general es necesario escribir nuevo código, sobre todo cuando se mueven a aplicaciones Nube Nativa y modelos basados en microservicios. Este enfoque puede ayudar a obtener beneficios que son difíciles de alcanzar en entornos de aplicación monolítica y *on-premises*.

La Tabla 1-1 describe los principales beneficios y razones para la elección de cada migración o enfoque de modernización.

Preparado para Infraestructura en la Nube	Preparado para DevOps en la nube	Optimizado para la Nube
<i>Migración directa</i>		<i>Modernizar/refactorizar/reescribir</i>
Punto de vista del procesamiento de la aplicación		
Aplicaciones desplegadas a VMs en Azure	Aplicaciones monolíticas contenerizadas o N-Tier desplegadas a VMs, Azure Service Fabric, o Azure Container Service (p.e. Kubernetes)	Microservicios contenerizados o aplicaciones comunes basadas en PaaS en Azure App Service, Azure Service Fabric, Azure Container Service (p.e. Kubernetes)
Punto de vista de los datos		
SQL o cualquier base de datos relacional en una VM	Azure SQL Database Managed Instance	Azure SQL Database, Azure Cosmos DB, u otra NoSQL
Ventajas		

<ul style="list-style-type: none"> • Sin necesidad de redefinir arquitecturas, ni código nuevo • Menor esfuerzo para una migración rápida • El mínimo común denominador soportado en Azure • Garantías de disponibilidad básica • Después de pasar a la nube, es más fácil modernizar aún más 	<ul style="list-style-type: none"> • Sin necesidad de redefinir arquitecturas, ni código nuevo • Los contenedores ofrecen menor esfuerzo incremental respecto a las máquinas virtuales • Despliegue mejorado y agilidad de DevOps al liberar entregas (<i>releases</i>) gracias a los contenedores • Mayor densidad y menores costes de despliegue • Portabilidad de aplicaciones y de dependencias • Con Azure Container Service (o Kubernetes) y Azure Service Fabric, proporciona alta disponibilidad y orquestación • Parcheo de Nodos/VM en Service Fabric • Flexibilidad para el destino del <i>host</i>: máquinas virtuales de Azure o conjuntos de escalado de VM, Azure Container Service (o Kubernetes), Service Fabric y opciones futuras basadas en contenedores. 	<ul style="list-style-type: none"> • Diseñar para la nube, refactorizar, nuevo código necesario • Enfoques de Microservicios nativos de la nube • Nuevas aplicaciones web, monolíticas, N-Tier, resilientes (resistentes a fallos) a la nube y optimizadas para la nube • Servicios completamente administrados • Parcheo automático • Optimizado para escalar • Optimizado para agilidad autónoma por subsistema • Construido en despliegue y DevOps • DevOps avanzado, como <i>slots</i> y estrategias de despliegue • Destinos de PaaS y orquestador: Azure App Service, Azure Container Service (o Kubernetes), Azure Service Fabric y futuros PaaS basados en contenedores.
Retos		
<ul style="list-style-type: none"> • Menor valor de la nube, que no sea un cambio en el gasto de operación o el cierre de centros de datos • Se gestiona muy poco: sin parches de SO o <i>middleware</i>; podría requerir de soluciones de infraestructura inmutables como Terraform, Spinnaker o Puppet 	<ul style="list-style-type: none"> • Contenerizar es un paso adicional en la curva de aprendizaje 	<ul style="list-style-type: none"> • Podría requerir la refactorización de una parte significativa del código o reescribir (mayor tiempo y presupuesto)

Tabla 1-1. Beneficios y retos de los caminos de modernización para aplicaciones y servicios .NET existentes

Tecnologías clave y arquitecturas por nivel de madurez

Las aplicaciones .NET Framework inicialmente comenzaron con la versión de .NET Framework 1.0, que fue lanzado a finales de 2001. A continuación, las empresas se movieron hacia versiones más recientes (por ejemplo, 2.0, 3.5 y 4.x .NET). La mayoría de estas aplicaciones se ejecutan en Windows Server e Internet Information Server (IIS), y utilizan una base de datos relacional, como SQL Server, Oracle, MySQL o cualquier otro RDBMS.

La mayoría de las aplicaciones .NET existentes podrían estar hoy en día basadas en .NET Framework 4.x, o incluso en .NET Framework 3.5, y utilizar Web Frameworks como ASP.NET MVC, ASP.NET Web Forms, ASP.NET Web API, Windows Communication Foundation (WCF), ASP.NET SignalR y páginas Web ASP.NET. Estas establecidas tecnologías .NET Framework dependen de Windows. Esta dependencia es importante considerar cuando simplemente se están migrando aplicaciones *legacy* (antiguas) y solo se van a realizar cambios mínimos en la infraestructura de la aplicación.

La Figura 1-2 muestra las tecnologías principales y los estilos de arquitectura utilizados para cada uno de los tres niveles de madurez en la nube:

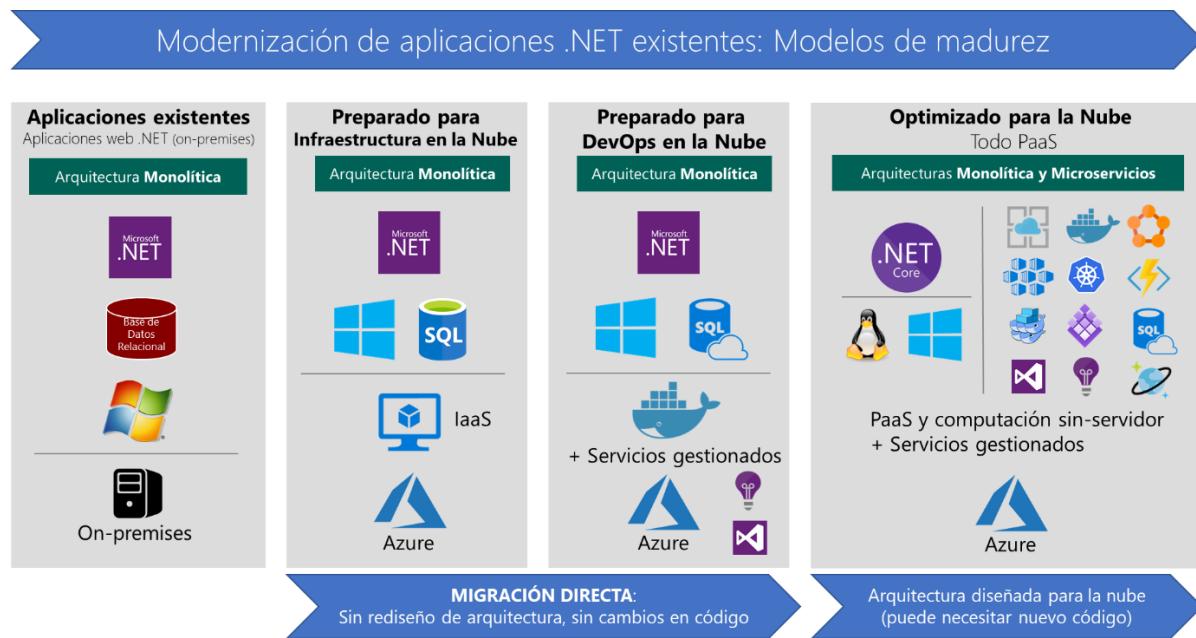


Figure 1-2. Tecnologías principales para cada nivel de madurez al modernizar aplicaciones web .NET existentes

La Figura 1-2 destaca los escenarios más comunes, pero existen muchas variaciones de tipo híbrido y mixtos que son posibles cuando hablamos de arquitectura. Por ejemplo, los modelos de madurez aplican no sólo a arquitecturas monolíticas en aplicaciones web existentes, sino también a las orientadas a servicios, las N-Tier, y otras variaciones en el estilo de la arquitectura.

Cada nivel de madurez en el proceso de modernización está asociado con las siguientes tecnologías y enfoques clave:

- **Preparado para Infraestructura en la Nube:** (reubicar o migración directa): Como primer paso, muchas organizaciones desean simplemente llevar a cabo una estrategia de migración a

la nube lo más rápidamente posible. En este caso, las aplicaciones simplemente son reubicadas (*rehosting*). La mayoría de los *rehosting* se pueden automatizar mediante el uso de [Azure Migrate](#), un servicio que proporciona la orientación, los puntos de vista, y los mecanismos necesarios para facilitar la migración a Azure, basado en herramientas como [Site Recovery](#) y [Azure Database Migration Service](#). Además, se puede configurar manualmente el *rehosting*, ayudandote a comprender los detalles de la infraestructura de tus activos cuando se mueven aplicaciones *legacy* a la nube. Por ejemplo, es posible mover aplicaciones a VMs en Azure, con muy pocas modificaciones—probablemente solo con cambios de configuración menores. La red en este caso es similar a un entorno *on-premises*, especialmente si se crean redes virtuales en Azure.

- **Preparado para DevOps en la Nube:** (migración directa mejorada): Este modelo trata de llevar a cabo algunas optimizaciones importantes en cuanto al despliegue, para conseguir ciertos beneficios significativos de la nube, sin cambiar la arquitectura principal de la aplicación. El paso fundamental aquí es añadir soporte a [Contenedores Windows](#) en las aplicaciones .NET Framework existentes. Este paso importante (contenerización) no requiere tocar el código, por lo que el esfuerzo general de la migración directa es muy liviano. Se pueden utilizar herramientas como [Image2Docker](#) o Visual Studio, con sus herramientas para [Docker](#). Visual Studio elige automáticamente los valores óptimos predeterminados para las aplicaciones ASP.NET y las imágenes de Contenedores Windows. Estas herramientas permiten un rápido proceso de desarrollo y agilidad en el despliegue de contenedores en Azure. La agilidad mejora cuando se despliega a múltiples entornos. A continuación, en el paso a producción, se pueden desplegar Contenedores Windows a orquestadores como [Azure Service Fabric](#) o [Servicio Contenedor de Azure](#) (Kubernetes, DC/OS o Swarm). Durante esta modernización inicial, también se pueden añadir mejoras desde la nube, tales como el seguimiento utilizando herramientas como [Azure Application Insights](#); Procesos CI/CD para los ciclos de vida de las aplicaciones con [Visual Studio Team Services](#); y muchos más servicios de recursos de datos disponibles en Azure. Por ejemplo, es posible modificar una aplicación web monolítica que fue desarrollada originalmente utilizando [Web Forms ASP.NET](#) tradicionales o [ASP.NET MVC](#), y ahora despliegarse mediante el uso de Contenedores Windows. Cuando se utilizan Contenedores Windows, también se deben migrar los datos a una [Azure SQL Database Managed Instance](#), todo sin cambiar la arquitectura principal de la aplicación.
- **Optimizado para la Nube:** Como se ha señalado, el objetivo final cuando se modernizan aplicaciones en la nube es basar los sistemas en plataformas PaaS como [Azure App Service](#). Las plataformas PaaS se enfocan a aplicaciones web modernas, y extienden las aplicaciones con nuevos servicios basados en [la informática sin servidores](#) y plataformas como [Azure Functions](#). El segundo y más avanzado escenario en este modelo de madurez es el de arquitecturas basadas en microservicios y [en aplicaciones Nativas para la Nube](#), que normalmente utilizan orquestadores como [Azure Service Fabric](#) o [Azure Container Service](#) (Kubernetes, DC/OS o Swarm). Estos orquestadores están hechos específicamente para aplicaciones basadas en microservicios y en contenedores múltiples. Todas estas aproximaciones (como microservicios y PaaS) requieren típicamente que se escriba nuevo código—código adaptado a plataformas PaaS específicas, o código que se alinea con arquitecturas específicas, como microservicios.

La Figura 1-3 muestra las tecnologías internas que se pueden utilizar para cada nivel de madurez:

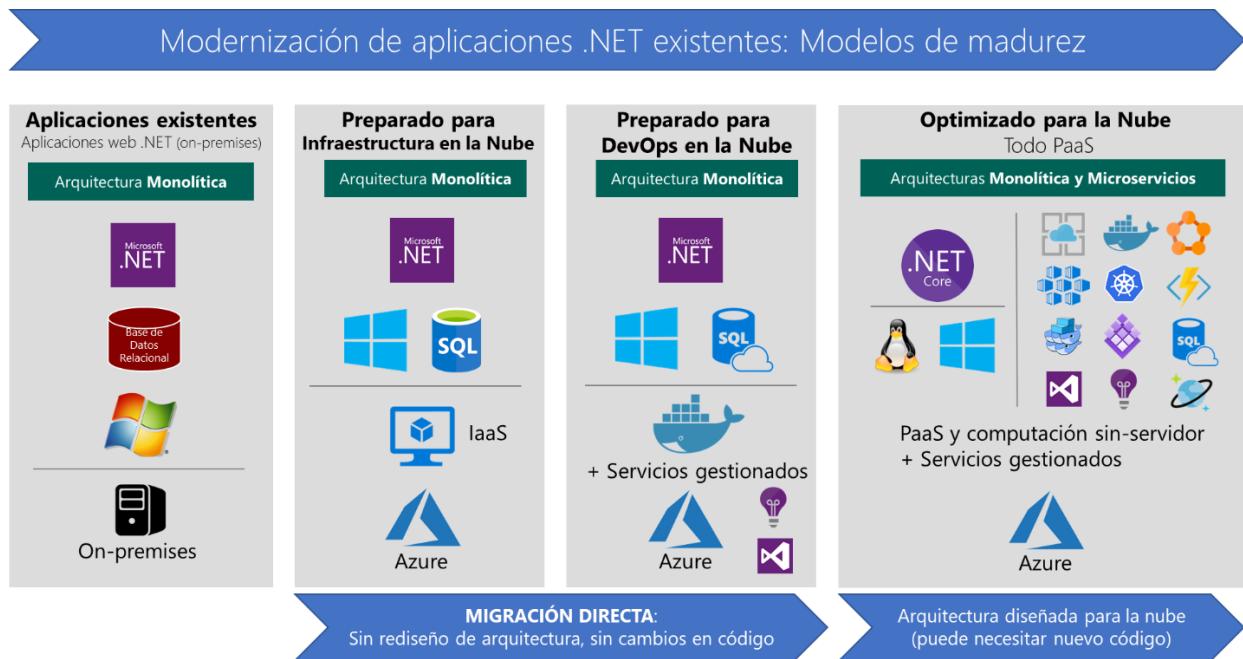


Figura 1-3. Tecnologías internas para cada nivel de madurez en la modernización

Escenarios de migración directa

Para migraciones directas o sencillas, hay que tener en cuenta que se pueden utilizar diferentes variaciones en los escenarios de cada aplicación. Si sólo se hace *rehosting* de la aplicación, es posible que se tenga un escenario como el que muestra la Figura 1-4, en el que se utilizan máquinas virtuales en la nube sólo para la aplicación y para el servidor de base de datos.



Figura 1-4. Ejemplo de un escenario IaaS puro en la nube

En el futuro, se podría tener una aplicación DevOps pura en la nube, que utilice elementos solo de ese nivel de madurez. O bien, es posible tener una aplicación en un estado intermedio con algunos elementos de aplicación Preparada para Infraestructura en la Nube y otros elementos de aplicación Preparada para DevOps en la Nube (un "escoger y elegir" o modelo mixto), como en la Figura de 1-5.



Figura 1-5. Ejemplo de escenario mixto, con base de datos en IaaS, DevOps, y contenedores

A continuación, como el escenario ideal de migración para muchas aplicaciones .NET Framework existentes, se pueden migrar a aplicaciones Preparadas para DevOps en la Nube, y así obtener grandes beneficios por poco trabajo. Esta aproximación también permite postergar su optimización para la nube a un posible paso futuro. La Figura 1-6 muestra un ejemplo.



Figura 1-6. Ejemplo de escenario de aplicación Preparada para DevOps en la Nube, con Contenedores Windows y servicios gestionados

Yendo más allá, se podrían extender las aplicaciones existentes Preparadas para DevOps en la Nube añadiendo algunos microservicios para escenarios específicos. Esto permitiría mover parcialmente las aplicaciones a nivel Nativo para la Nube dentro del modelo de Optimizada para la Nube, que no es el foco de la presente guía.

Lo que esta guía no cubre

Esta guía cubre un subconjunto específico de escenarios de ejemplo, como se muestra en la Figura 1-7. Esta guía se centra sólo en escenarios de migración directa o sencilla, y en última instancia, en el modelo Preparado para DevOps en la Nube. En el modelo de Preparado para DevOps en la Nube, una aplicación de .NET Framework se moderniza mediante Contenedores Windows, además de componentes adicionales como monitorización y procesos CI/CD. Cada componente es fundamental para desplegar aplicaciones a la nube, más rápido y con agilidad.

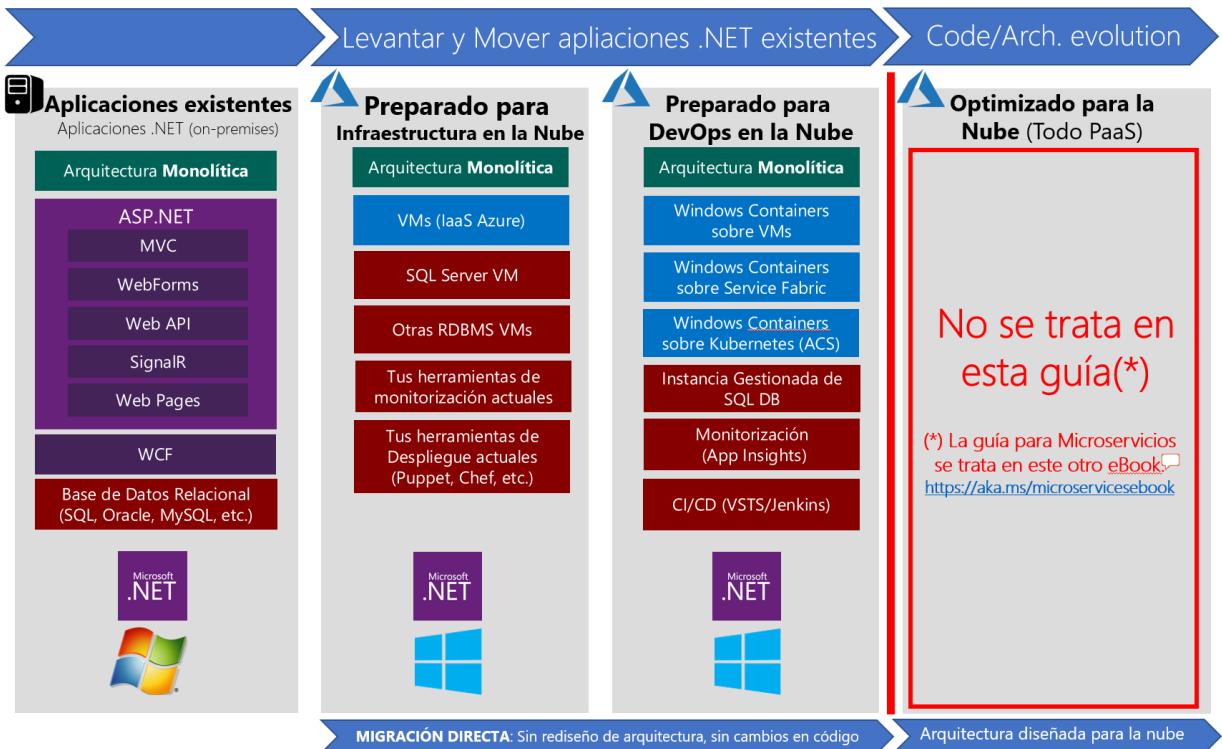


Figura 1-7. Migrar directamente y modernización inicial a aplicación Preparada para DevOps en la Nube

El objeto de esta guía es concreto. Explicaremos el camino que se puede tomar para lograr migrar directamente aplicaciones .NET existentes, sin necesidad de redefinir arquitecturas, y sin cambios en el código. Finalmente, mostraremos cómo preparar una aplicación para DevOps en la nube.

Esta guía no enseña cómo trabajar con aplicaciones Nativas para la Nube, como por ejemplo la forma de evolucionar a arquitecturas de microservicios. Para redefinir la arquitectura de las aplicaciones o para crear nuevas aplicaciones que se basan en microservicios, consultar el libro electrónico [.NET Microservices: Architecture for containerized .NET applications](#).

Recursos adicionales

- **Containerized Docker application lifecycle with Microsoft platform and tools** (eBook descargable): <https://aka.ms/dockerlifecyclebook>
- **.NET Microservices: Architecture for containerized .NET applications** (eBook descargable): <https://aka.ms/microservicesebook>
- **Architecting modern web applications with ASP.NET Core and Azure** (eBook descargable): <https://aka.ms/webappebook>

Quién debe usar esta guía

Escribimos esta guía para desarrolladores y arquitectos de soluciones que quieren modernizar aplicaciones ASP.NET existentes basadas en .NET Framework, para mejorar la agilidad en el lanzamiento y entrega (*releasing*) de aplicaciones.

También puede resultar útil esta guía para personas responsables en la toma de decisiones técnicas, tales como arquitectos empresariales o responsables/directores de desarrollo que simplemente

necesitan tener una visión general de los beneficios del uso de Contenedores Windows, y del despliegue a la nube utilizando Microsoft Azure.

Como usar esta guía

Esta guía se centra en el “por qué” —¿Por qué puedo querer modernizar mis aplicaciones, y cuáles son los beneficios específicos que se obtienen por el uso de Contenedores Windows al mover mis aplicaciones a la nube?. El contenido de los primeros capítulos de la guía están diseñados para arquitectos y responsables de decisiones técnicas que quieren una visión general, pero que no necesitan centrarse en la implementación y la técnica, ni descripciones detalladas tipo paso-a-paso.

El último capítulo de esta guía introduce múltiples tutoriales paso-a-paso que se centran en escenarios de despliegue específicos. En esta guía, ofrecemos versiones más cortas de estos tutoriales, para resumir los escenarios y poner de relieve sus ventajas. Los tutoriales completos profundizan en los detalles de configuración e implementación, y se publican como un conjunto de [posts wiki](#) en el mismo repositorio público de [GitHub](#) donde residen las aplicaciones de ejemplo relacionadas (se discute en la siguiente sección). El último capítulo y los tutoriales paso-a-paso en GitHub serán de mayor interés para desarrolladores y arquitectos que quieren centrarse en los detalles de implementación.

Aplicaciones ejemplo para modernizar aplicaciones *legacy*: eShopModernizing

El repositorio [eShopModernizing](#) en GitHub ofrece dos aplicaciones de ejemplo que simulan aplicaciones web *legacy* (heredadas) monolíticas. Una de las aplicaciones web está desarrollada usando ASP.NET MVC; la segunda aplicación web está desarrollada mediante el uso de formularios Web ASP.NET. Ambas aplicaciones web se basan en .NET Framework tradicional. Estas aplicaciones de ejemplo no utilizan .NET Core o Core ASP.NET, ya que se supone que son aplicaciones existentes/*legacy* de .NET Framework para ser modernizadas.

Ambas aplicaciones de ejemplo tienen una segunda versión, con el código modernizado, y son bastante claras. La diferencia más importante entre las dos versiones de las aplicaciones es que las segundas utilizan Contenedores Windows como la elección para despliegue. Las segundas versiones también tienen algunos añadidos, como el uso de Azure Storage Blobs para la gestión de imágenes, Azure Active Directory para la gestión de la seguridad y Azure Application Insights para el seguimiento y auditoría de las aplicaciones.

¡Envíanos tus comentarios!

Hemos escrito esta guía para ayudarte a entender las opciones existentes para la mejora y la modernización de aplicaciones web .NET existentes. Tanto la guía como las aplicaciones ejemplo relacionadas continúan evolucionando. ¡Agradecemos tus comentarios! Si tienes comentarios sobre cómo esta guía puede ser más útil, por favor envíalos a dotnet-architecture-ebooks-feedback@service.microsoft.com.

Migrar de manera directa aplicaciones .NET existentes a Azure IaaS (Preparadas para Infraestructura en la Nube)

Visión: Como primer paso, para reducir la inversión en *on-premises* y el coste global de hardware y mantenimiento de redes, simplemente hacer *rehosting* de las aplicaciones existentes en la nube.

Antes de entrar en *cómo* migrar las aplicaciones existentes a la plataforma Azure de infraestructura como servicio (IaaS), es importante analizar las razones por las que te gustaría migrar directamente a IaaS en Azure. El escenario en este nivel de madurez de modernización es en esencia comenzar a utilizar VMs en la nube, en lugar de seguir utilizando la actual infraestructura *on-premises*.

Otro punto a analizar es *por qué* se quiere migrar a nube IaaS pura en lugar de simplemente a añadir servicios gestionados avanzados en Azure. Es necesario determinar qué casos podrían requerir IaaS en primera instancia.

La Figura 2-1 posiciona aplicaciones Preparadas para Infraestructura en la Nube en los diferentes niveles de madurez en modernización:

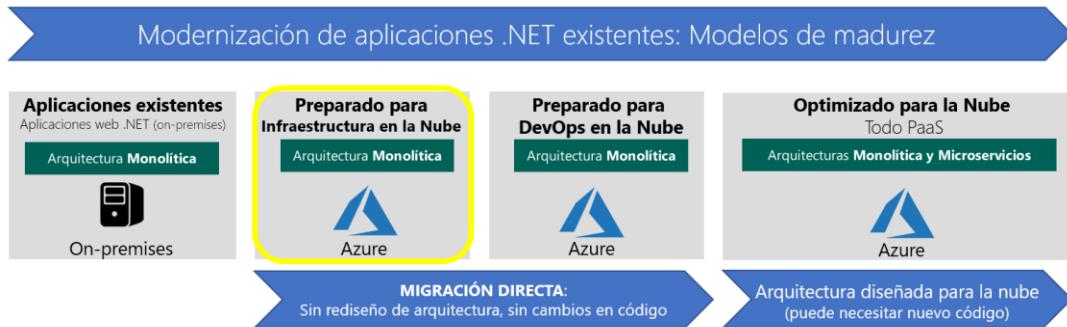


Figura 2-1. Posicionamiento de aplicaciones Preparadas para Infraestructura en la Nube

Por qué migrar aplicaciones web .NET existentes a Azure IaaS

La razón principal para migrar a la nube, incluso a un nivel IaaS inicial, es para lograr reducir costes. Mediante el uso de más servicios de infraestructura gestionados, la organización puede reducir su inversión en mantenimiento de hardware, servidores o aprovisionamiento y despliegue de máquinas virtuales, y la gestión de infraestructuras.

Después de tomar la decisión de mover las aplicaciones a la nube, la razón principal por la cual optar por IaaS en lugar de otras opciones más avanzadas como PaaS es simplemente que el entorno IaaS puede resultar más familiar. Pasar a un entorno que es similar al actual consigue que los entornos *on-premises* tengan una menor curva de aprendizaje, lo que hace que sean el camino más rápido a la nube.

Sin embargo, tomar el camino más rápido a la nube no significa que se vaya a obtener el mayor beneficio de tener las aplicaciones ejecutándose en la nube. Cualquier organización obtendrá los beneficios más significativos de una migración a la nube en los ya introducidos niveles de madurez Preparado para DevOps en la Nube y PaaS (Optimizado para la Nube).

También se considera evidente que las aplicaciones serán más fáciles de modernizar y redefinir en el futuro si ya se estarán ejecutando en la nube, incluso en IaaS. Esto es cierto en parte, debido a que la migración de datos de la aplicación a la nube ya se habrá llevado a cabo. Además, la organización habrá conseguido las habilidades necesarias para trabajar en la nube, y se habrá transformado para poder operar en una "cultura cloud".

Cuando migrar a IaaS en lugar de a PaaS

En las siguientes secciones, se discute sobre las aplicaciones Preparadas para DevOps en la Nube, que se basan principalmente en plataformas y servicios PaaS. Estas aplicaciones ofrecen los mayores beneficios de migrar a la nube.

Si el objetivo es simplemente mover las aplicaciones existentes a la nube, en primer lugar, se deben identificar las aplicaciones existentes que requerirán modificaciones sustanciales para ejecutarse en Azure App Service. Estas aplicaciones deben ser las primeras candidatas.

Después, si no se desea o aún no se pueden mover a Contenedores Windows y orquestadores como Azure Service Fabric o Kubernetes, entonces es cuando se utilizarían simplemente VMs planas (IaaS).

Sin embargo, hay que tener en cuenta que configurar correctamente, securizar y mantener máquinas virtuales requiere mucho más tiempo y experiencia en TI en comparación con el uso servicios PaaS en Azure. Si se está pensando en máquinas virtuales de Azure, hay que asegurarse de que se tiene en cuenta el esfuerzo de mantenimiento requerido para parchear, actualizar y administrar un entorno de máquinas virtuales. Las máquinas virtuales en Azure son IaaS.

Uso de Azure Migrate para analizar y migrar aplicaciones existentes a Azure

Migrar a la nube no tiene por qué ser difícil. Pero a muchas organizaciones les cuesta arrancar — tratando de obtener una visibilidad profunda del entorno y de las estrictas interdependencias entre aplicaciones, cargas de trabajo y datos. Sin esa visibilidad, puede ser difícil planificar el camino hacia adelante. Sin información detallada sobre lo que se requiere para una migración con éxito, no se pueden tener las conversaciones correctas dentro de la organización. No se sabe lo suficiente sobre los posibles beneficios del coste de migrar, o si las cargas de trabajo podrían simplemente migrarse directamente o requerirán una repetición significativa de trabajo para migrar con éxito. No es de extrañar que muchas organizaciones duden.

[Azure Migrate](#) es un nuevo servicio que proporciona la orientación, ideas y mecanismos necesarios para facilitar la migración a Azure. Azure Migrate proporciona:

- Descubrimiento y evaluación de máquinas virtuales *on-premises*
- Mapeo de dependencias internas para descubrir aplicaciones *multi-tier* con garantías
- Dimensionamiento inteligente a máquinas virtuales de Azure
- Informes de compatibilidad con pautas para solucionar posibles problemas
- Integración con Azure Database Management Service para la detección y migración de bases de datos

Azure Migrate genera la confianza de que las cargas de trabajo se pueden migrar con un impacto mínimo al negocio y ejecutarse según lo esperado en Azure. Con las herramientas y la orientación adecuadas, se puede obtener el máximo rendimiento de la inversión al mismo tiempo que se garantiza que se satisfagan las necesidades críticas de rendimiento y fiabilidad.

La Figura 2-2 muestra el mapeo de dependencias para todas las conexiones de servidores y aplicaciones llevadas a cabo por Azure Migrate.

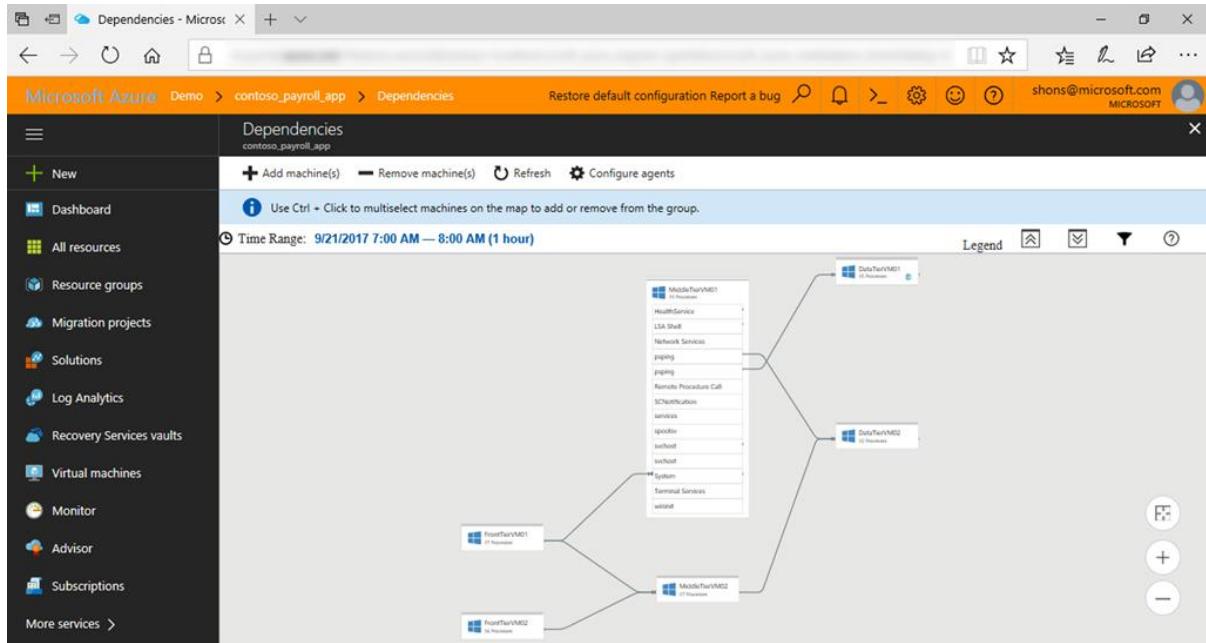


Figura 2-2. Mapeo de dependencias en Azure Migrate

Usar Azure Site Recovery para migrar las VMs existentes a Azure VMs

Como parte de la herramienta *end-to-end* [Azure Migrate](#), [Azure Site Recovery](#) se puede utilizar para migrar fácilmente aplicaciones web a máquinas virtuales en Azure. Se puede utilizar Site Recovery para replicar VMs *on-premises* y servidores físicos a Azure, o para replicar a otro sitio secundario *on-premises*. Incluso se puede replicar una carga de trabajo que se ejecuta en una máquina virtual Azure soportada, en una VM Hyper-V *on-premises*, en una VM de VMware, o en un servidor físico Windows o Linux. La replicación en Azure elimina el coste y la complejidad de mantener un centro de datos secundario.

Site Recovery también está diseñado específicamente para entornos híbridos que son en parte *on-premises* y en parte en Azure. Site Recovery ayuda a garantizar la continuidad del negocio gracias a que mantiene las aplicaciones que se ejecutan en VMs y en servidores físicos *on-premises* disponibles si un sitio se cae. Replica cargas de trabajo que se ejecutan en VMs y servidores físicos para que permanezcan disponibles en una ubicación secundaria si el sitio principal no está disponible. Recupera cargas de trabajo al sitio principal cuando está en funcionamiento de nuevo.

La Figura 2-3 muestra la ejecución de múltiples migraciones de VMs mediante el uso de Azure Site Recovery.

The screenshot shows the 'Replicated items' blade in the Azure portal. At the top, it says 'ContosoRecoveryServices' and has 'Refresh', 'Replicate', and 'Columns' buttons. Below that, it says 'Last refreshed at: 10/13/2016, 2:59:17 PM'. A message box says 'Finished loading data from service.' There is a search bar with 'Filter items...'. The main table has columns: NAME, HEALTH, STATUS, ACTIVE LOCATION, and REPLICATION POLICY. It lists several items:

NAME	HEALTH	STATUS	ACTIVE LOCATION	REPLICATION POLICY
AWSServer2012	OK	Unplanned failover comp...	Microsoft Azure	AWSReplicationPolicy
AWSWordPressRedHat	OK	Protected	AWSGATEWAY	AWSReplicationPolicy
FabrikamMarketing	OK	Unplanned failover comp...	Microsoft Azure	ContosoReplicationPolicy
FabrikamFinance	OK	Protected	CONTOSOGATEWAY	ContosoReplicationPolicy
▶ ContosoReplicationGr...	-	-	-	-

A context menu is open for the last row ('▶ ContosoReplicationGr...'). The menu options are: Pin to dashboard, Unplanned Failover, Test Failover, Change PIT, Commit, Complete Migration (which is highlighted with a red box), Re-protect, Resynchronize, Error Details, and Delete.

Figura 2-3. Azure Site Recovery con múltiples migraciones

Recursos adicionales

- **Hoja de datos de Azure Migrate**
https://aka.ms/azurermigration_datasheet
- **Azure Migrate**
<http://azurermigrationcenter.com/>
- **Migrar a Azure con Site Recovery**
<https://docs.microsoft.com/es-es/azure/site-recovery/site-recovery-migrate-to-azure>
- **Descripción general del servicio Azure Site Recovery**
<https://docs.microsoft.com/es-es/azure/site-recovery/site-recovery-overview>
- **Migración de VMs en AWS a VMs de Azure**
<https://docs.microsoft.com/es-es/azure/site-recovery/site-recovery-migrate-aws-to-azure>

Migrar bases de datos relacionales a Azure

Visión: Azure ofrece la migración de base de datos más completa.

En Azure, se pueden migrar servidores de bases de datos directamente a máquinas virtuales IaaS (pura migración directa), o se pueden migrar a SQL Azure Database, para obtener beneficios adicionales. Azure SQL Database ofrece una instancia gestionada y opciones completas de base de datos como servicio (DBaaS). En la Figura 3-1 se muestran los múltiples caminos para la migración de bases de datos relacionales.

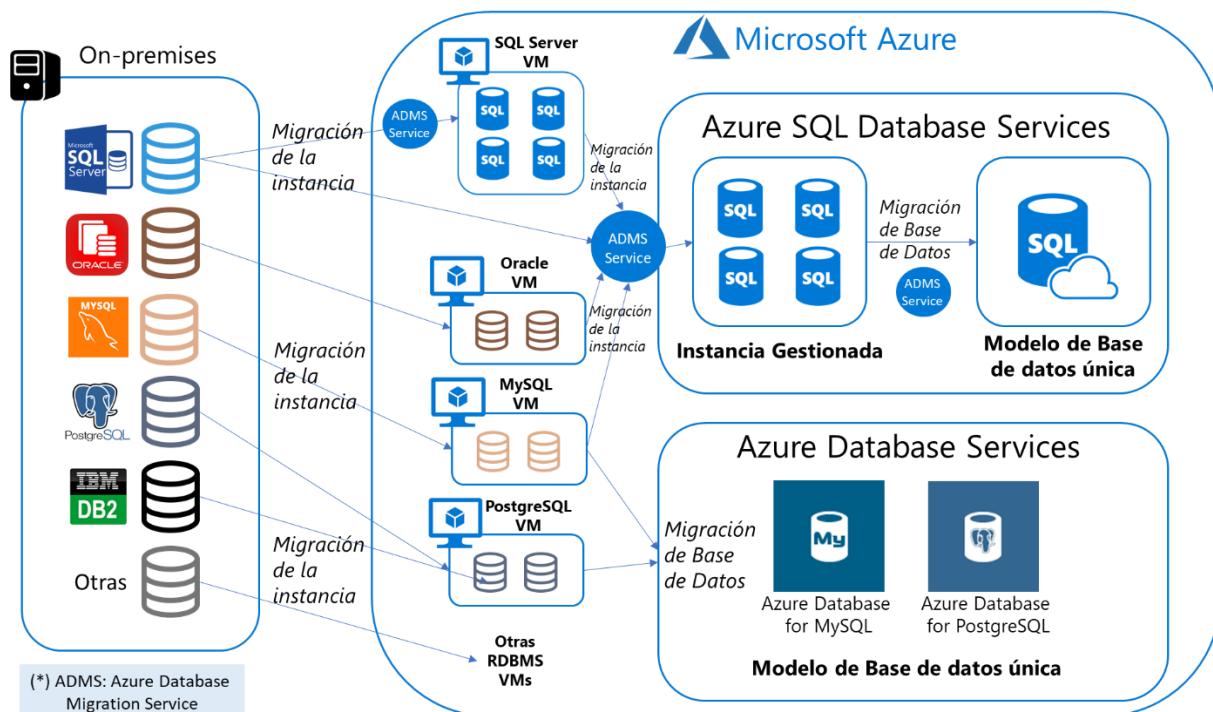


Figura 3-1. Caminos para migración de bases de datos a Azure

Cuando migrar a Azure SQL Database Managed Instance

En la mayoría de los casos, Azure SQL Database Managed Instance será la mejor opción a considerar cuando se migran datos a Azure. Si se están migrando bases de datos de SQL Server y se necesita cerca del 100% de garantías de que no se va a tener que volver a redefinir la arquitectura de la aplicación o realizar cambios en los datos o el código de acceso a los datos, es recomendable seleccionar la opción de una Managed Instance de Azure SQL Database.

Azure SQL Database Managed Instance es la mejor opción si se tienen requisitos adicionales para la funcionalidad a nivel de instancia de SQL Server, o los requisitos de aislamiento más allá de las características proporcionadas en una base de datos Azure SQL estándar (solo modelo de base de datos). Esta última es la opción más orientada a PaaS, pero no ofrece las mismas características que un SQL Server tradicional. La migración podría sufrir fricciones.

Por ejemplo, una organización que ha realizado grandes inversiones en capacidades a nivel de instancia de SQL Server se beneficiaría de la migración a una SQL Managed Instance. Ejemplos de capacidades a nivel de instancia de SQL Server incluyen la integración de Common Language Runtime (CLR), SQL Server Agent, y consultas *cross-database* (contra múltiples bases de datos). El soporte para estas características no está disponible en una base de datos estándar de Azure SQL (un modelo de única base de datos).

Una organización que opera en una industria altamente regulada, y que necesita mantener aislamiento por motivos de seguridad, también podría beneficiarse de la elección del modelo de SQL Managed Instance.

Managed Instance en Azure SQL Database tiene las siguientes características:

- Seguridad de aislamiento a través Azure Virtual Network
- Volumen de compatibilidad en la aplicación, con las siguientes características:
 - Agente SQL Server y SQL Server Profiler
 - Consultas y referencias cruzadas entre bases, SQL CLR, replicación, captura de cambios en datos (CDC) y Service Broker
- Tamaño de base de datos de hasta 35 TB
- Mínimo tiempo de inactividad en migración, gracias a estas características:
 - Azure Database Migration Service
 - Backup y restore nativos, y envío de *log* (registro).

Con estas capacidades, cuando migramos bases de datos de aplicación existentes a Azure SQL Database, el modelo de Managed Instance ofrece cerca del 100% de los beneficios de Paas para SQL Server. Managed Instance es un entorno SQL Server donde se continúan utilizando capacidades a nivel de instancia, sin cambiar el diseño de la aplicación.

Managed Instance es probablemente la mejor opción para las empresas que actualmente utilizan SQL Server, y que requieren de flexibilidad en la seguridad de su red en la nube. Es como tener una red privada virtual para las bases de datos SQL.

Cuándo migrar a Azure SQL Database

Como se mencionó, la base de datos Azure SQL estándar es una DBaaS relacional completamente gestionada. Azure SQL Database actualmente gestiona millones de bases de datos en producción, a través de 38 centros de datos, en todo el mundo. Es compatible con una amplia gama de aplicaciones y cargas de trabajo, desde la gestión de datos en transacciones sencillas, hasta manejar las aplicaciones con el mayor uso intensivo de datos, aplicaciones de misión crítica que requieren procesamiento de datos avanzado a escala global.

Debido a sus completas características PaaS y su mejor precio—y en última instancia, un menor coste—se debe escoger la base de datos estándar Azure SQL como la “opción por defecto” si se tiene una aplicación que utiliza las características básicas, y estándar de una base de datos SQL, y no hay características adicionales a nivel de instancia. Las características de SQL Server como la integración con SQL CLR, el SQL Server Agent, y las consultas *cross-database* no son compatibles con SQL Azure Database estándar. Estas características sólo están disponibles en el modelo de Azure SQL Database Managed Instance.

Azure SQL Database es el único servicio de base de datos inteligente en la nube que está construido para los desarrolladores de aplicaciones. Es también el único servicio de base de datos en la nube que escala en el momento, sin tiempo de inactividad, para facilitar el despliegue eficiente de aplicaciones *multitenant* (multiples clientes/empresas con datos segregados por aplicación). En última instancia, Azure SQL Azure permite dedicar el tiempo a innovar, y agiliza el plazo de comercialización de las aplicaciones. Se pueden construir aplicaciones seguras, y conectarlas a bases de datos SQL utilizando los lenguajes y plataformas que se quieran.

Azure SQL Database ofrece los siguientes beneficios:

- Servicios de inteligencia (aprendizaje automático) internos, que aprenden y se adaptan a cada aplicación
- Aprovisionamiento de bases de datos bajo demanda
- Amplio rango de oferta, para todas las cargas de trabajo
- SLA de disponibilidad del 99.99%, con cero mantenimiento
- Geo-replicación y servicios de restauración para la protección de datos
- Función de restauración a un momento dado (Azure SQL Azure Point in Time Restore)
- Compatibilidad con SQL Server 2016, incluyendo entornos híbridos y migración

La versión estándar de Azure SQL Database está más cerca de PaaS que una Azure SQL Database Managed Instance. Se debe tratar de utilizar, si es posible, porque se obtendrán mayores beneficios gracias a una nube administrada. Sin embargo, Azure SQL Database tiene algunas diferencias clave comparándose con instancias SQL Server *on-premises* normales. Dependiendo de los requisitos de base de datos concretos de cada aplicación, y las necesidades y políticas de cada empresa, podría no ser la mejor opción cuando se está planeando una migración a la nube.

Cuándo mover una RDBMS de origen a una VM (IaaS)

Una de las opciones de migración es mover el sistema original de gestión de bases de datos relacionales (RDBMS), incluyendo Oracle, IBM DB2, MySQL, PostgreSQL o SQL Server, a un servidor similar que se ejecute en una Azure VM. Si existen aplicaciones que requieren una migración rápida a la nube con un mínimo de cambios o sin ningún cambio en absoluto, una migración directa a IaaS en la nube podría ser la opción más equilibrada. Podría no ser la mejor manera para beneficiarse de todas las ventajas de la nube, pero es probablemente la ruta inicial más rápida.

Actualmente, Microsoft Azure soporta hasta [348 servidores de bases de datos diferentes](#) desplegados como IaaS VMs. Incluyendo los RDBMSs más populares como SQL Server, Oracle, MySQL, PostgreSQL e IBM DB2, y muchas otras bases de datos NoSQL como MongoDB, Cassandra, DataStax, MariaDB, y Cloudera.

Hay que tener en cuenta que a pesar de que mover un RDBMS a una VM en Azure puede ser la manera más rápida para migrar los datos a la nube (porque es IaaS), este enfoque requiere una inversión significativa en el personal de IT (administradores de bases de datos e IT pros). Este personal debe ser capaz de preparar y administrar la alta disponibilidad, la recuperación ante desastres y el parcheo de SQL Server. Este contexto también necesita un entorno personalizado, con permisos de administración total.

Cuándo migrar a SQL Server como una VM (IaaS)

Puede haber algunos casos en los que todavía sea necesario migrar a SQL Server como una VM estándar. Un escenario ejemplo puede ser si es necesario utilizar SQL Server Reporting Services. En la mayoría de los casos, sin embargo, Azure SQL Database Managed Instance puede proporcionar todo lo necesario para migrar desde servidores SQL *on-premises*, por lo que la migración a una VM de SQL Server debe ser el último recurso a probar.

Uso de Azure Database Migration Service para migrar bases de datos relacionales a Azure

Se puede utilizar Azure Database Migration Service para migrar bases de datos relacionales como SQL Server, Oracle y MySQL a Azure, tanto si la base de datos destino es Azure SQL Database, como Azure SQL Database Managed Instance, o SQL Server en una Azure VM.

El flujo de trabajo automatizado, con los informes de evaluación, sirven de guía a través de los cambios que se deben llevar a cabo antes migrar la base de datos. Cuando se esté listo, el servicio migrará la base de datos de origen a Azure.

Cuando se cambia un RDBMS de origen, puede ser necesario que se tengan que repetir pruebas. También es posible que se necesiten cambiar sentencias SQL o código Object-Relational Mapping (ORM - Mapeo de bases de datos relacionales a objetos) en la aplicación, dependiendo de los resultados de las pruebas.

Si se cuenta con cualquier otra base de datos (por ejemplo, IBM DB2) y se opta por un enfoque de migración directa o sencilla, es posible que se quieran seguir utilizando las bases de datos como IaaS VMs en Azure, a menos que se esté dispuesto a realizar una migración de datos más compleja. Una migración de datos más compleja requerirá esfuerzo adicional, ya que se estaría migrando a un tipo de base de datos diferente con un nuevo esquema y diferentes librerías de programación.

Para aprender cómo migrar las bases de datos utilizando Azure Database Migration Service, consulta [Get to the cloud faster with Azure SQL Database Managed Instance and Azure Database Migration Service](#).

Recursos adicionales

- **Elegir una opción de SQL Server en la nube: Base de datos (PaaS) SQL de Azure o SQL Server en máquinas virtuales de Azure (IaaS)**
<https://docs.microsoft.com/azure/sql-database/sql-database-paas-vs-sql-server-iaas>
- **Get to the cloud faster with Azure SQL DB Managed Instance and Database Migration Service**
<https://channel9.msdn.com/Events/Build/2017/P4008>
- **Migración de una base de datos de SQL Server a una Base de datos SQL en la nube**
<https://docs.microsoft.com/azure/sql-database/sql-database-cloud-migrate>
- **Azure SQL Database**
<https://azure.microsoft.com/services/sql-database/?v=16.50>
- **SQL Server en Virtual Machines**
<https://azure.microsoft.com/services/virtual-machines/sql-server/>

Migrar de manera directa aplicaciones .NET existentes a aplicaciones Preparadas para DevOps en la Nube

Visión: Migrar de manera directa aplicaciones .NET Framework existentes a aplicaciones Preparadas para DevOps en la Nube para mejorar drásticamente la agilidad en el despliegue, lo hace posible lanzar aplicaciones más rápido y así reducir los costes de despliegue.

Para aprovechar los beneficios de la nube y nuevas tecnologías como los contenedores, se deben modernizar al menos parcialmente las aplicaciones existentes .NET. En último término, la modernización de aplicaciones empresariales disminuirá el coste total de propiedad.

Modernizar parcialmente una aplicación no significa necesariamente llevar a cabo una migración completa y redefinir su arquitectura. Se pueden modernizar inicialmente las aplicaciones existentes mediante el uso de un proceso de migración directa, que es fácil y rápido. Se puede mantener el código base actual, escrito en versiones .NET Framework existentes, con cualquier

dependencia de Windows y IIS. La Figura 4-1 pone de relieve cómo las aplicaciones Preparadas para DevOps en la Nube se posicionan en los modelos de madurez de modernización de aplicaciones en Azure.

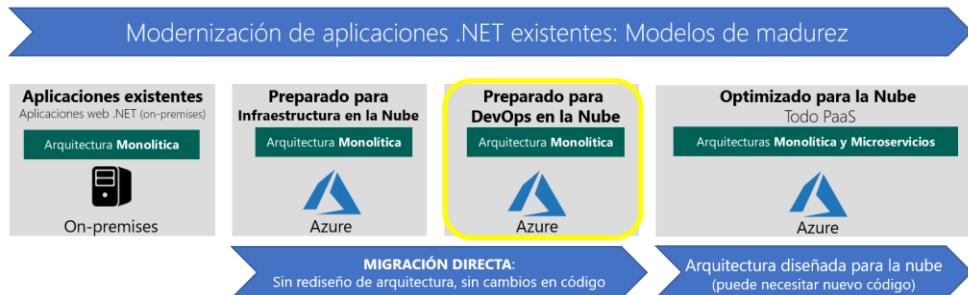


Figura 4-1. Posicionamiento de aplicaciones Preparadas para DevOps en la Nube

Razones para migrar directamente aplicaciones .NET existentes a aplicaciones Preparadas para DevOps en la Nube

Con una aplicación Preparada para DevOps en la Nube, es posible entregar rápida y repetidamente aplicaciones fiables a los clientes. Se gana en agilidad y fiabilidad esenciales, delegando la complejidad operativa de la aplicación a la plataforma.

Si no es posible poner rápidamente las aplicaciones en el mercado, en el momento en el que se lancen, el mercado al que se habían orientado habrá evolucionado. Podría ser demasiado tarde, sin importar lo buena que sea la arquitectura de la aplicación o su diseño. Se podría estar fallando, o no llegando al potencial máximo, por no poder sincronizar la entrega de la aplicación con las necesidades del mercado.

La necesidad de innovación continua del negocio lleva a los equipos de desarrollo y operaciones al límite. La única manera de lograr la agilidad necesaria para la innovación continua del negocio es mediante la modernización de las aplicaciones con tecnologías como contenedores y principios específicos de aplicaciones Preparadas para DevOps en la Nube.

En resumidas cuentas, cuando una organización construye y gestiona aplicaciones Preparadas para DevOps en la Nube, puede poner antes soluciones en manos de sus clientes, y aportar nuevas ideas al mercado cuando son relevantes.

Principios y procedimientos de aplicaciones Preparadas para DevOps en la Nube

Las mejoras de la nube se centran sobre todo en la satisfacción de dos objetivos: Reducir costes, y mejorar el crecimiento del negocio mejorando su agilidad. Estos

objetivos se logran simplificando procesos y reduciendo la fricción cuando se liberan *releases* y lanzan aplicaciones.

Una aplicación está Preparada para DevOps en la Nube si se puede—de una manera ágil—desarrollar autónomamente respecto a otras aplicaciones *on-premises*, y luego entregar, desplegar, auto-escalar, monitorizar y resolver sus incidencias en la nube.

La clave es la *agilidad*. No se puede lanzar una aplicación ágilmente a menos que se reduzca a un mínimo absoluto cualquier incidencia en el despliegue a producción y en el entorno de desarrollo/pruebas. Los contenedores (específicamente, Docker, es un estándar de facto) y los servicios gestionados han sido diseñados específicamente para este propósito.

Para lograr agilidad, se necesitan también procesos DevOps automatizados basados en CI/CD que liberen aplicaciones a plataformas escalables en la nube. Plataformas CI/CD (como Visual Studio Team Services o Jenkins) para desplegar a plataformas escalables y flexibles en la nube (como Azure Service Fabric o Kubernetes) son las tecnologías clave para lograr agilidad en la nube.

En la siguiente lista se describen los principios fundamentales y prácticas para las aplicaciones Preparadas para DevOps en la Nube. Es importante recalcar que se pueden adoptar todos o sólo algunos de estos principios, en un enfoque progresivo o incremental:

- **Contenedores.** Los contenedores dan la capacidad de incluir dependencias de la aplicación con la propia aplicación. El uso de contenedores reduce significativamente el número de incidencias que pueden surgir cuando se despliega a entornos de producción o se prueba en entornos de *staging*. En última instancia, los contenedores mejoran la agilidad en la entrega de aplicaciones.
- **Nube resiliente y escalable.** La nube proporciona una plataforma administrada, elástica, escalable y resiliente (resistente a fallos). Estas características son fundamentales para obtener beneficios en los costes y lanzar aplicaciones de alta disponibilidad y fiables de manera continua. Los servicios gestionados como las bases de datos gestionadas, el caché administrado como servicio (CaaS), y el almacenamiento gestionado son piezas fundamentales para aliviar los costes de mantenimiento de las aplicaciones.
- **Monitorización.** No se puede tener una aplicación fiable sin tener una buena manera de detectar y diagnosticar excepciones y problemas de rendimiento. Es necesario disponer de información profunda a través de la gestión del rendimiento de las aplicaciones y el análisis instantáneo.
- **Cultura DevOps y de entrega continua.** Adoptar las prácticas de DevOps en la nube requiere de un cambio cultural en el que los equipos ya no trabajan en silos independientes. Los procesos CI/CD sólo son posibles cuando hay un aumento de la colaboración entre los equipos de desarrollo y de operaciones de IT, gracias a los contenedores y las herramientas CI/CD.

La Figura 4-2 muestra los principales pilares opcionales de una aplicación Preparada para DevOps en la Nube. Cuantos más pilares se implementen, más preparada estará la aplicación para cumplir las expectativas de los clientes.

Aplicaciones Preparadas para DevOps en la Nube

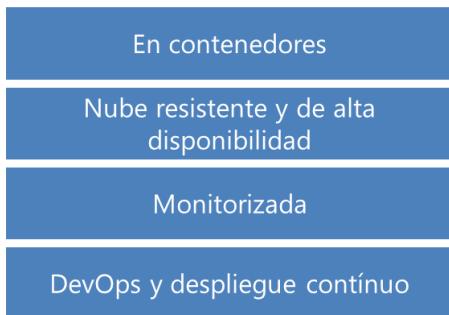


Figura 4-2. Principales pilares de una aplicación Preparada para DevOps en la Nube

Para resumir, una aplicación Preparada para DevOps en la Nube supone un enfoque para la construcción y gestión de aplicaciones que se aprovecha del modelo de computación en la nube, gracias a una combinación de contenedores, infraestructura gestionada en la nube, técnicas de aplicación resiliente, monitorización, entrega continua y DevOps, todo ello sin la necesidad de redefinir la arquitectura ni recodificar las aplicaciones existentes.

Una organización puede adoptar estas tecnologías y enfoques de forma gradual. No se tienen por qué abarcar todos ellos, ni todos a la vez. Se pueden adoptar de forma incremental, dependiendo de las prioridades de la empresa y las necesidades de los usuarios.

Beneficios de una aplicación Preparada para DevOps en la Nube

Se pueden obtener los siguientes beneficios mediante la conversión de una aplicación existente a una aplicación Preparada para DevOps en la Nube (sin volver a definir la arquitectura o su codificación):

- **Menores costes, porque la infraestructura administrada es soportada por el proveedor de la nube.** Las aplicaciones Preparadas para DevOps en la Nube consiguen los beneficios de la nube mediante el uso de la elasticidad implícita de la nube, auto-escalado, y alta disponibilidad. Los beneficios no solo están relacionados con las características de procesamiento (VM y contenedores), sino que también dependen de los recursos en la nube, como DBaaS, CaaS, y cualquier infraestructura que una aplicación pueda necesitar.
- **Aplicación resiliente e infraestructura.** Al migrar a la nube, es necesario asumir fallos transitorios; fallos que ocurrirán en la nube. Además, la infraestructura de nube y el hardware son "reemplazables", lo que aumenta las posibilidades de tiempo de inactividad transitoria. Al mismo tiempo, las capacidades inherentes de la nube y ciertas técnicas de desarrollo de aplicaciones que implementan resiliencia y recuperación automática, hacen que sea mucho más fácil recuperarse de fallos inesperados en la nube.

- **Visión más profunda sobre el rendimiento de las aplicaciones.** Las herramientas de monitorización de la nube como Azure Application Insights permiten analizar información para la gestión del estado o salud de la aplicación, *logging* (registro) y envío de notificaciones. Los *logs* de auditoría hacen que las aplicaciones sean más fáciles de depurar y auditar. Esto es fundamental para tener una aplicación en la nube confiable.
- **Portabilidad de la aplicación, con despliegues ágiles.** Los contenedores (ya sean Linux o Windows basados en contenedores Docker Engine) ofrecen la mejor solución para evitar aplicaciones bloqueadas en la nube. Mediante el uso de contenedores, Docker hosts, y orquestadores multi-nube, se puede mover fácilmente de un entorno o nube a otra. Los contenedores eliminan la fricción que se produce normalmente en los despliegues a cualquier entorno (*stage/test/producción*).

Todos estos beneficios al final son la clave en la reducción de costes en el ciclo de vida *end-to-end* de las aplicaciones.

En las siguientes secciones, se explicarán estos beneficios en mayor detalle, vinculándose a tecnologías específicas.

Tecnologías Microsoft en aplicaciones Preparadas para DevOps en la Nube

La siguiente lista describe las herramientas, tecnologías, y soluciones que se reconocen como requisitos para las aplicaciones Preparadas para DevOps en la Nube. Se pueden comenzar a desarrollar aplicaciones Preparadas para DevOps en la Nube de manera selectiva o poco a poco, en función de las prioridades.

- **Infraestructura de la nube:** La infraestructura que provee la plataforma de computación, sistema operativo, red y almacenamiento. Microsoft Azure se posiciona en este nivel.
- **Runtime:** Esta capa proporciona el entorno para que la aplicación se ejecute. Si se están utilizando contenedores, esta capa por lo general se basa en [Docker Engine](#), ejecutándose en servidores Linux o en máquinas Windows. ([Los Contenedores Windows](#) comenzaron a ser compatibles con Windows Server 2016. Los Contenedores Windows son la mejor opción para aplicaciones .NET Framework existentes que se ejecutan en Windows).
- **Nube gestionada:** Cuando se elige una opción de nube gestionada, es posible evitar el gasto y la complejidad de gestionar y soportar la infraestructura subyacente, máquinas virtuales, parches de sistema operativo y configuración de red. Si se elige migrar mediante el uso de IaaS, se es responsable de todas estas tareas, así como de los gastos asociados. En la opción de nube gestionada, se gestionan sólo las aplicaciones y servicios que se desarrollean. El proveedor de servicios en la nube normalmente gestiona todo lo demás. Algunos ejemplos de servicios gestionados en Azure incluyen [Azure SQL Database](#), [Azure Redis Cache](#), [Azure Cosmos DB](#), [Azure Storage](#), [Azure Database for MySQL](#), [Azure Database for PostgreSQL](#), [Azure Active Directory](#), y servicios gestionados de computación como [VM scale sets](#), [Azure Service Fabric](#), [Azure App Service](#), y [Azure Container Service](#).

- **Desarrollo de aplicaciones:** Se puede elegir entre muchos lenguajes cuando se construyen aplicaciones que se ejecutan en contenedores. En esta guía, nos centramos en [.NET](#), pero, se pueden desarrollar aplicaciones basadas en contenedores mediante el uso de otros lenguajes, como Node.js, Python, Spring/Java o GoLang.
- **Monitorización, telemetría, *logging*, y auditorias:** La capacidad de monitorizar y auditar aplicaciones y contenedores que se ejecutan en la nube es crítica para cualquier aplicación Preparada para DevOps en la Nube. [Azure Application Insights](#) y [Microsoft Operations Management Suite](#) son las principales herramientas de Microsoft que proporcionan monitorización y auditorias para aplicaciones Preparadas para DevOps en la Nube.
- **Aprovisionamiento:** Las herramientas de automatización ayudan a la provisión de infraestructura y a desplegar una aplicación en múltiples entornos (producción, testing, *staging*). Se pueden utilizar herramientas como Chef y Puppet para administrar la configuración de una aplicación y su entorno. Esta capa también se puede implementar usando enfoques más simples y directos. Por ejemplo, se puede desplegar directamente mediante el uso de la herramienta de interfaz de línea de comandos de Azure (CLI Azure), y luego utilizar los procesos de despliegue continuo y gestión de *releases* de [Visual Studio Team Services](#).
- **Ciclo de vida de la aplicación:** [Visual Studio Team Services](#) y otras herramientas, como Jenkins, son servidores de automatización de *builds* (compilaciones) que le ayudan a implementar procesos CI/CD, incluyendo la gestión de *releases*.

Las siguientes secciones de este capítulo, y las guías paso-a-paso relacionadas, se centran específicamente en los detalles sobre la capa de ejecución (Contenedores Windows). La guía describe las formas en que se pueden desplegar Contenedores Windows en VMs de Windows Server 2016 (y versiones posteriores). También cubre las capas de orquestación más avanzadas, como Azure Service Fabric, Kubernetes, y Azure Container Service. La creación de capas de orquestación es un requisito fundamental para modernizar aplicaciones .NET Framework existentes (basadas en Windows), como las aplicaciones Preparadas para DevOps en la Nube.

Las aplicaciones monolíticas *pueden* ser aplicaciones Preparadas para DevOps en la Nube

Es importante destacar que las aplicaciones monolíticas (aplicaciones que no se basan en microservicios) *pueden* ser aplicaciones Preparadas para DevOps en la Nube. Se pueden construir y operar aplicaciones monolíticas que se aprovechan del modelo de computación en la nube mediante el uso de una combinación de contenedores, entrega continua, y DevOps. Si una aplicación monolítica existente es adecuada para los objetivos del negocio, se puede modernizar y convertir en una aplicación Preparada para DevOps en la Nube.

Del mismo modo, si las aplicaciones monolíticas pueden ser aplicaciones Preparadas para DevOps en la Nube, otras arquitecturas, más complejas, como N-Tier también pueden modernizarse como aplicaciones Preparadas para DevOps en la Nube.

¿Qué pasa con las aplicaciones optimizadas para la nube?

A pesar de que las aplicaciones optimizadas para la nube y las [aplicaciones Nativas para la Nube](#) no son el foco principal de esta guía, es útil tratar de entender este nivel de madurez de modernización, para distinguirlo del nivel de las aplicaciones Preparadas para DevOps en la Nube.

La Figura 4-3 posiciona las aplicaciones optimizadas para la nube en los diferentes niveles de madurez en la modernización de aplicaciones:

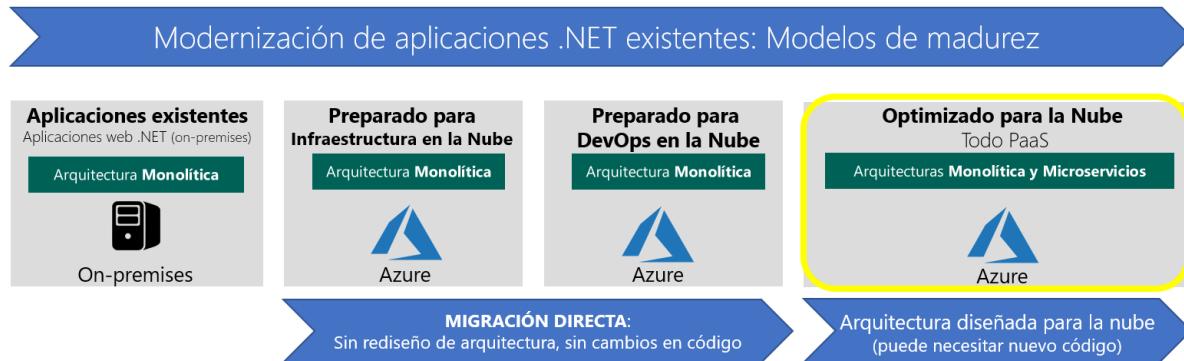


Figura 4-3. Posicionamiento de aplicaciones Optimizadas para la Nube

El nivel de madurez de modernización Optimizada para la Nube por lo general requiere nuevas inversiones en desarrollo. El paso al nivel de optimización para la nube normalmente es impulsado por la necesidad del negocio de modernizar aplicaciones tanto como sea posible para reducir costes, aumentar la agilidad y aventajar a la competencia. Estos objetivos se logran maximizando el uso de la nube PaaS. Esto significa no sólo el uso de servicios PaaS como DBaaS, CaaS, y el almacenamiento como servicio (STaaS), sino también migrando las aplicaciones y servicios a una plataforma de computación PaaS como Azure App Service, o mediante el uso de orquestadores.

Este tipo de modernización por lo general requiere refactorizar o escribir nuevo código que esté optimizado para las plataformas PaaS en la nube (como para Azure App Service). Incluso podría requerir definir arquitecturas específicamente para el entorno de la nube, sobre todo si se está moviendo a modelos de aplicación Nativa para la Nube que se basan en microservicios. Este es un factor clave diferencial en comparación con el nivel de Preparada para DevOps en la Nube, que no requiere redefinir arquitecturas y ni escribir código nuevo.

En algunos casos más avanzados, se pueden crear [aplicaciones Nativas para la Nube](#) basadas en arquitecturas de microservicios, que pueden evolucionar ágilmente y escalar a límites que serían difíciles de lograr en una arquitectura monolítica desplegada en un entorno *on-premises*.

La Figura 4-4 muestra el tipo de aplicaciones que se pueden desplegar cuando se utiliza el modelo de optimización para la nube. Existen dos opciones básicas—aplicaciones web modernas y aplicaciones Nativas para la Nube.

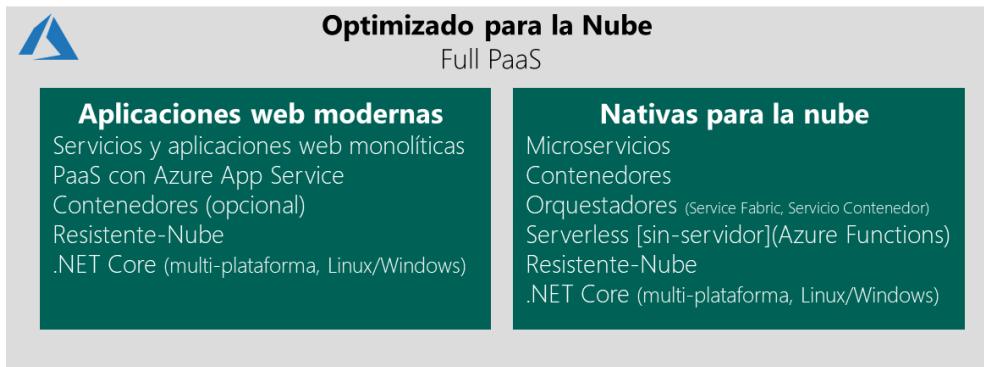


Figura 4-4. Tipos de aplicaciones en el nivel Optimizado para la Nube

Se pueden extender aplicaciones modernas web básicas y aplicaciones nativas en la nube añadiendo otros servicios, como la inteligencia artificial (IA), el aprendizaje automático (ML), y el IoT. Es posible utilizar cualquiera de estos servicios para extender cualquiera de los posibles enfoques de optimización para la nube.

La diferencia fundamental en las aplicaciones al nivel de optimización para la nube se encuentra en la arquitectura de la aplicación. Las [aplicaciones Nativas para la Nube](#) son, por definición, aplicaciones que se basan en microservicios. Las aplicaciones Nativas para la Nube requieren de arquitecturas, tecnologías y plataformas especiales, en comparación con las aplicaciones web monolíticas o las aplicaciones tradicionales N-Tier.

Crear nuevas aplicaciones que no utilizan microservicios también tiene sentido. Hay muchos escenarios nuevos y también modernos en los que un enfoque basado en microservicios podría exceder las necesidades. En algunos casos, es posible que se quiera crear una aplicación web monolítica más simple, o añadir servicios grandes a una aplicación N-Tier. En estos casos, aún se pueden utilizar capacidades de la nube PaaS como las ofrecidas por Azure App Service. Se seguirá reduciendo el trabajo de mantenimiento hasta el límite.

También, porque se desarrolla nuevo código en escenarios optimizados para la nube (para una aplicación completa o para subsistemas parciales), cuando se crea nuevo código, se deben utilizar las nuevas versiones de .NET ([.NET Core](#) y [ASP.NET Core](#), en particular). Esto es especialmente cierto si se crean microservicios y contenedores porque .NET Core es un framework ligero y rápido. Se conseguirá poco impacto en el uso de memoria y un comienzo rápido en el uso de contenedores, y las aplicaciones se ejecutarán con alto rendimiento. Este enfoque encaja perfectamente con los requisitos de microservicios y contenedores, y se obtendrán los beneficios de un *framework* multiplataforma—siendo capaz de ejecutar la misma aplicación en Linux, Windows Server y Mac (Mac para entornos de desarrollo).

Aplicaciones nativas con aplicaciones Optimizadas para la Nube

Nativo para la Nube es un estado más avanzado o maduro para aplicaciones de gran tamaño y de misión crítica. Las aplicaciones Nativas para la Nube por lo general requieren arquitecturas y diseños que son creados desde cero, en lugar de mediante la modernización de aplicaciones existentes. La diferencia clave entre una aplicación Nativa para la Nube y una aplicación web más simple Optimizada para la Nube desplegada en PaaS, es la recomendación de utilizar arquitecturas de microservicios en un enfoque en la Nube Nativa. Las aplicaciones optimizadas para la nube también pueden ser aplicaciones web monolíticas o N-Tier desplegadas a servicios PaaS en la nube, como Azure App Service.

El [Twelve-Factor App](#) (una colección de patrones que están estrechamente relacionados con los enfoques de microservicios) también se considera un requisito para las arquitecturas de aplicaciones [Nativas para la Nube](#).

La fundación [Cloud Native Computing Foundation \(CNCF\)](#) es un promotor primordial de principios para aplicaciones Nativas para la Nube. Microsoft es un [miembro de la CNCF](#).

Para una definición ejemplo y para obtener más información sobre las características de las aplicaciones Nativas para la Nube, ver el artículo de [How to architect and design cloud-native applications](#). Para una guía específica de Microsoft acerca de cómo implementar aplicaciones Nativas para la Nube, consultar [.NET microservices: Architecture for containerized .NET applications](#).

El hecho más importante a considerar si se migra una aplicación completa al modelo de [Nube Nativa](#) es que se debe modificar la arquitectura a una basada en microservicios. Evidentemente, esto requiere de una inversión significativa en desarrollo, debido al gran proceso de refactorización involucrado. Esta opción se elige generalmente para aplicaciones de misión crítica que necesitan nuevos niveles de escalabilidad y mantener su agilidad a largo plazo. Sin embargo, se podría comenzar a un movimiento hacia Nube Nativa añadiendo microservicios para algunos escenarios nuevos, y eventualmente refactorizar la aplicación completamente como microservicios. Este es un enfoque gradual y supone la mejor opción para algunos escenarios.

¿Qué hay sobre los microservicios?

Comprender los microservicios y cómo funcionan es importante cuando se están considerando aplicaciones nativas en la organización.

La arquitectura de microservicios supone una aproximación avanzada que se puede utilizar para aplicaciones que se crean a partir de cero o cuando se evolucionan aplicaciones existentes (ya sea *on-premises* o aplicaciones Preparadas para DevOps en la Nube) hacia aplicaciones nativas en la nube. Se puede comenzar añadiendo algunos microservicios a las aplicaciones existentes para aprender acerca de los paradigmas de los nuevos microservicios. Pero está claro que se necesita definir una arquitectura y codificar, especialmente para este tipo de enfoque de arquitectura.

Sin embargo, los microservicios no son obligatorios para cualquier aplicación nueva o moderna. Los microservicios no son una “barita mágica”, y no son la única ni mejor manera para crear todas las aplicaciones. Cómo y cuándo se utilizan microservicios dependerá del tipo de aplicación que se necesita construir.

La arquitectura de microservicios se está convirtiendo en el enfoque preferido para aplicaciones distribuidas de misión crítica, grandes o complejas que se basan en múltiples subsistemas independientes en forma de servicios autónomos. En una arquitectura basada en microservicios, una aplicación está construida como un conjunto de servicios que se pueden desarrollar de forma independiente, probada, versionada, desplegada, y escalada. Esto puede incluir cualquier base de datos autónoma por cada microservicio.

Para un estudio más detallado sobre una arquitectura de microservicios, que se puede implementar mediante el uso de .NET Core, ver el libro electrónico [.NET microservices: Architecture for containerized .NET applications](#). Esta guía también está disponible [online](#).

Pero incluso en escenarios en los que los microservicios ofrecen potentes capacidades—despliegue independiente, fuertes límites en subsistemas, y diversidad tecnológica—también plantean nuevos retos. Estos retos están relacionados con el desarrollo de aplicaciones distribuidas, tales como modelos de datos fragmentados e independientes; alcanzar una comunicación resiliente entre microservicios; la necesidad de consistencia eventual; y la complejidad operacional. Los microservicios introducen un mayor nivel de complejidad en comparación con las aplicaciones monolíticas tradicionales.

Debido a la complejidad de una arquitectura de microservicios, sólo los escenarios específicos y ciertos tipos de aplicaciones son adecuados para aplicaciones basadas en microservicios. Estas incluyen aplicaciones grandes y complejas que tienen varios subsistemas en evolución. En estos casos, vale la pena invertir en una arquitectura de software más compleja, para mantener una mayor agilidad persistente a largo plazo y un mantenimiento más eficiente de las aplicaciones. Pero para escenarios de menor complejidad, es posible que sea mejor continuar con un enfoque de aplicación monolítica o enfoques más simples N-Tier.

Como nota final, aun a riesgo de ser repetitivo acerca este concepto, no se debe plantear como un “todo o nada en absoluto” el uso microservicios en las aplicaciones. Se pueden extender y evolucionar las aplicaciones monolíticas existentes añadiendo nuevos, y pequeños escenarios basados en microservicios. No es necesario construir desde cero para comenzar a trabajar con un enfoque de arquitectura de microservicios. De hecho, se recomienda la evolución de aplicaciones monolíticas o N-Tier mediante la incorporación de nuevos escenarios. Eventualmente, se puede descomponer la aplicación en componentes autónomos o microservicios. Se puede comenzar evolucionando las aplicaciones monolíticas en dirección a microservicios, paso a paso.

Cuando utilizar Azure App Service para modernizar aplicaciones .NET existentes

Cuando se modernizan aplicaciones ASP.NET Web al nivel de madurez Optimizado para la Nube, debido a que las aplicaciones web se han desarrollado utilizando .NET Framework, las principales dependencias están en Windows y, muy probablemente, en Internet Information Server (IIS). Se pueden usar y desplegar aplicaciones basadas en Windows y en IIS mediante un despliegue directo a [Azure App Service](#) o por la contenerización primero de la aplicación mediante Contenedores Windows. Si se conteneriza, se deben desplegar las aplicaciones ya sea a *Hosts* de Contenedores Windows (basados en VM) o a un clúster de Azure Service Fabric que soporte Contenedores Windows.

Cuando se utilizan Contenedores Windows, se obtienen todos los beneficios de la contenerización. También se aumenta la agilidad en el lanzamiento y despliegue de la aplicación, y se reduce la fricción causada por incidencias en el entorno (*staging*, dev/test, producción). En las siguientes secciones, se entrará en más detalles acerca de los beneficios que se obtienen al utilizar contenedores.

En el momento de la redacción de esta guía, Azure App Service no es compatible con Contenedores Windows. Si soporta contenedores para Linux. Por lo tanto, la próxima pregunta podría ser: "¿Cómo elegir entre Azure App Service y de Contenedores Windows?"

Básicamente, si Azure App Service funciona para una aplicación y no existe ninguna dependencia personalizada o de servidor que bloquee el camino a utilizar App Service, se debe migrar la aplicación .NET existente a App Service. Esa es la manera más fácil, y más eficaz de mantener la aplicación. En Azure, la aplicación también requerirá de un mantenimiento más sencillo gracias a los beneficios de la infraestructura PaaS, como DBaaS, CaaS, y STaaS.

Sin embargo, si la aplicación tiene dependencias personalizadas o de servidor que no son compatibles con Azure App Service, puede que se tengan que considerar las opciones basadas en Contenedores Windows. Ejemplos de dependencias de servidor o personalizadas incluyen dependencias de software de terceros, o de un fichero .msi que necesita ser instalado en el servidor pero que no se soporta en Azure App Service. Otro ejemplo es cualquier otra configuración de servidor que no se soporte en Azure App Service, como el uso de ensamblados en la Global Assembly Cache (GAC) o componentes COM/COM+. Gracias a las imágenes de los Contenedores Windows, se pueden incluir las dependencias personalizadas en la misma "unidad de despliegue".

Como alternativa, se pueden refactorizar las áreas de la aplicación que no sean compatibles con Azure App Service. Dependiendo del volumen de trabajo que requiera la refactorización, habría que evaluar cuidadosamente si vale la pena hacerlo.

Beneficios de moverse a Azure App Service

Azure App Service es una solución PaaS completamente gestionada que facilita la construcción de aplicaciones web respaldadas por procesos de negocio. Cuando se utiliza App Service, se evitan los costes de gestión de infraestructura asociados con la actualización y mantenimiento de aplicaciones web *on-premises*. En concreto, se eliminan los costes de hardware y licenciamiento que requieren ejecutar aplicaciones web *on-premises*.

Si una aplicación web es adecuada para ser migrada a Azure App Service, el principal beneficio es la poca cantidad de tiempo que se necesita para mover la aplicación. App Service proporciona un entorno muy fácil por el que comenzar.

Azure App Service es la mejor opción para la mayoría de las aplicaciones web, ya que es la solución PaaS más simple que se puede utilizar para ejecutar aplicaciones web. El despliegue y la gestión están integrados en la plataforma, los sitios escalan de forma rápida para soportar altas cargas de tráfico, y los gestores de balanceo y tráfico integrados proporcionan alta disponibilidad.

Incluso la monitorización de las aplicaciones web resulta sencilla, a través de Azure Application Insights. Application Insights viene gratis con App Service, y no requiere escribir código especial en la aplicación. Sólo hay que ejecutar su aplicación web en App Service, y se dispondrá de un sistema de monitorización, sin trabajo adicional.

Con App Service, también se pueden utilizar directamente muchas aplicaciones open-source (de código abierto) desde Azure Web Application Gallery (como WordPress o Umbraco), o se puede crear un nuevo sitio mediante el uso del framework y las herramientas que se elijan, como ASP.NET. Los Webjobs de App Service hacen que sea fácil agregar procesamiento de tareas en *background* para aplicaciones App Service web.

Las ventajas clave de la migración de aplicaciones web mediante el uso de la característica Web Apps de Azure App Service incluyen las siguientes:

- Escala automática para satisfacer la demanda durante las horas punta y reducir los costes durante momentos de tranquilidad.
- Copias de seguridad automáticas del sitio para proteger los cambios y datos.
- Alta disponibilidad y resiliencia de la plataforma Azure PaaS.
- *Slots* de despliegue para entornos de desarrollo y *staging*, y para *testing* de múltiples diseños de sitios.
- Balanceo de carga y protección de Denegación de Servicio Distribuida (DDoS).
- Gestión de tráfico para dirigir a los usuarios al despliegue geográfico más cercano.

A pesar de que App Service puede ser la mejor opción para aplicaciones web nuevas, sin embargo, para las aplicaciones existentes, App Service podría ser la mejor opción solo si las dependencias de las aplicaciones están soportadas en App Service.

Recursos adicionales

- **Análisis de compatibilidad para Azure App Service**
<https://www.migratetoazure.net/Resources>

Beneficios de pasar a Contenedores Windows

El principal beneficio de utilizar de Contenedores Windows es que se obtiene una experiencia de despliegue más fiable y mejorada, en comparación con las aplicaciones no contenerizadas. Además, tener las aplicaciones modernizadas con contenedores deja efectivamente a la aplicación preparada para muchas otras plataformas y nubes que soporten Contenedores Windows. Las ventajas de cambiar a Contenedores Windows están cubiertas en mayor detalle en las siguientes secciones.

Los principales entornos de procesamiento en Azure (en disponibilidad general a partir de mediados de 2017) que soportan Contenedores Windows son Azure Service Fabric y *hosts* de Contenedores Windows básicos (Windows Server 2016 VMs). Estos entornos son los principales escenarios de infraestructura cubiertos en esta guía.

También se pueden desplegar Contenedores Windows a otros orquestadores, como Kubernetes, Docker Swarm, o DC/OS. En la actualidad (otoño de 2017), estas plataformas están en *preview* en Azure Container Service para el uso de Contenedores Windows.

Como desplegar aplicaciones existentes .NET a Azure App Service

La característica Web Apps de Azure App Service es una plataforma de computación completamente gestionada que está optimizada para alojamiento de sitios y aplicaciones web. Esta solución PaaS de Microsoft Azure permite centrarse en la lógica de negocio, mientras que Azure se encarga de la infraestructura para ejecutar y escalar las aplicaciones.

Validar sitios y migrar a App Service con Azure App Service Migration Assistant

Cuando se crea una nueva aplicación en Visual Studio, mover la aplicación a App Service por lo general es una tarea sencilla. Sin embargo, si se está planeando migrar una aplicación existente a App Service, lo primero que es necesario evaluar es si todas las dependencias de la aplicación son compatibles con App Service. Esto incluye dependencias como el sistema operativo del servidor o cualquier otro software de terceros que esté instalado en el mismo.

Se puede utilizar [Azure App Service Migration Assistant](#) para analizar los sitios y luego migrar a partir de servidores web Windows y Linux a App Service. Como parte de la migración, la herramienta crea las web apps y bases de datos necesarias sobre Azure, publica el contenido, y publica la base de datos.

Azure App Service Migration Assistant soporta la migración a la nube desde IIS ejecutándose en Windows Server. App Service soporta Windows Server 2003 y versiones posteriores.

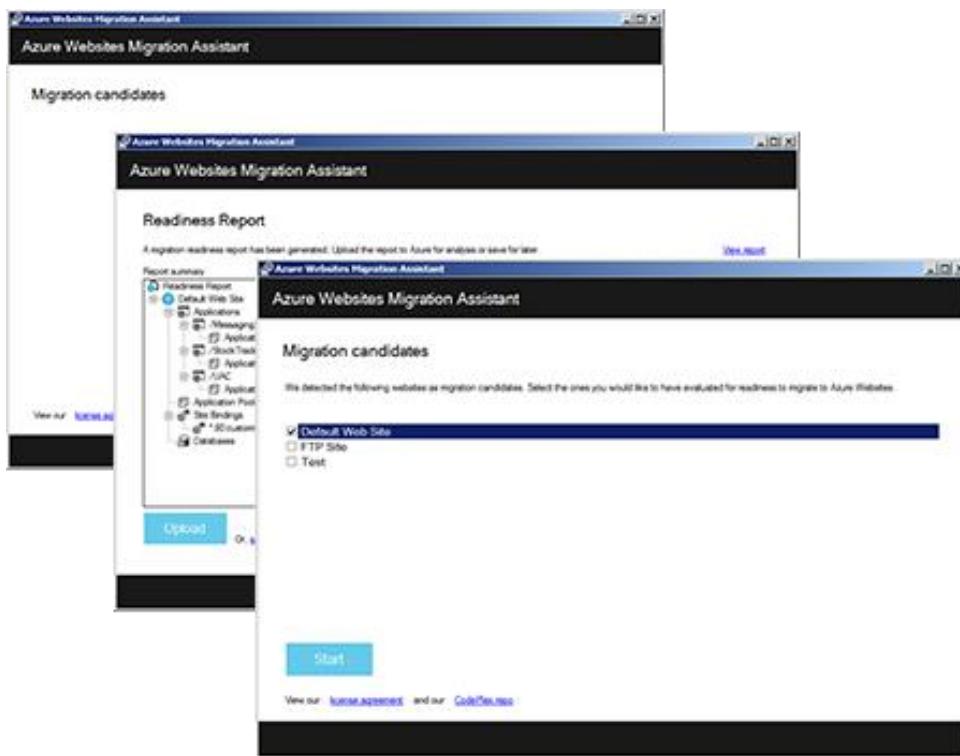


Figure 4-5. Using Azure App Service Migration Assistant

App Service Migration Assistant es una herramienta que mueve sitios web desde servidores de Internet a la nube de Azure.

Después de que un sitio web es migrado a App Service, el sitio tendrá todo lo que necesita para funcionar de manera segura y eficiente. Los sitios se configuran y ejecutan automáticamente en servicio PaaS de la nube de Azure (App Service).

La herramienta de migración de App Service puede analizar los sitios web e informar sobre su compatibilidad para moverse a App Service. Si el análisis es satisfactorio para el usuario, se puede dejar al App Service Migration Assistant migrar el contenido, datos, y ajustes. Si un sitio no es suficientemente compatible, la herramienta de migración dirá qué es lo que se necesita ajustar para que funcione.

Recursos adicionales

- **Azure App Service Migration Assistant**
<https://www.migratetoazure.net/>

Despliegue de aplicaciones .NET existentes como Contenedores Windows

Los despliegues basados en Contenedores Windows aplican a aplicaciones Optimizadas para la Nube, aplicaciones Nativas para la Nube, y aplicaciones Preparadas para DevOps en la Nube.

En esta guía, en las siguientes secciones, nos centramos en el uso de Contenedores Windows para aplicaciones Preparadas para DevOps en la Nube, cuando se mueven de manera directa aplicaciones .NET existentes.

¿Qué son los contenedores? (Linux o Windows)

Los contenedores son una manera de envolver una aplicación en su propio paquete aislado. En su contenedor, la aplicación no se ve afectada por aplicaciones o procesos que existen fuera del mismo. Todo de lo que depende la aplicación para funcionar como un proceso, y con éxito, está en el interior del contenedor. Dondequiera que el contenedor se mueva, los requisitos de la aplicación se satisfarán, en términos de dependencias directas, ya está empaquetada con todo lo que necesita para funcionar (dependencias de bibliotecas, *runtimes*, etc.) .

La principal característica de un contenedor es que hace que el entorno sea el mismo a través de diferentes despliegues debido a que el propio contenedor viene con todas las dependencias que necesita. Esto significa que se puede depurar la aplicación en una máquina, y luego desplegarla en otra, con el mismo entorno garantizado.

Un contenedor es una instancia de una imagen de contenedor. Una imagen de contenedor es una manera de empaquetar una aplicación o servicio (como una foto instantánea), y luego desplegarlo de una manera fiable y reproducible. Se podría decir que Docker no es sólo una tecnología—es también una filosofía y un proceso.

Como los contenedores se vuelven diariamente más comunes, se están convirtiendo en la “unidad de despliegue” generalizada en la industria.

Beneficios de los contenedores (Docker Engine en Linux o Windows)

La construcción de aplicaciones mediante el uso de contenedores—que también podrían ser definidos como bloques de construcción ligeros—ofrece un aumento significativo en la agilidad a la hora de construir, lanzar, y ejecutar cualquier aplicación, a través de cualquier infraestructura.

Con contenedores, se puede llevar cualquier aplicación desde desarrollo hasta producción con poco o sin ningún cambio en el código, gracias a la integración de Docker a través de herramientas para desarrolladores de Microsoft, sistemas operativos, y la nube.

Cuando se despliega a VMs, es probable que ya se cuente con un método para el despliegue de aplicaciones ASP.NET. Es probable, sin embargo, que el método consista en varios pasos manuales o procesos automatizados complejos mediante el uso de una herramienta de despliegue como Puppet, o similar. Es posible que se necesiten llevar a cabo tareas como la modificación de elementos de configuración, copiar el contenido de la aplicación entre servidores, ejecutar programas de instalación

interactivos basados en instaladores .msi, y a continuación las pruebas. Todos esos pasos en el despliegue le añaden tiempo y riesgo. Se obtendrán fallos cada vez que una dependencia no esté presente en el entorno de destino.

En Contenedores Windows, el proceso de empaquetado de aplicaciones está completamente automatizado. Los Contenedores Windows se basan en la plataforma Docker, que ofrece actualizaciones automáticas y *rollbacks* (deshacer cambios) para despliegues de contenedores. La principal mejora que se obtiene al usar el motor Docker es que se crean imágenes, que son como instantáneas de una aplicación, con todas sus dependencias. Las imágenes son imágenes Docker (una imagen de contenedor de Windows, en este caso). Las imágenes ejecutan aplicaciones ASP.NET en contenedores, sin volver al código fuente. La instantánea del contenedor se convierte en la unidad de despliegue.

Un número muy elevado de organizaciones están contenerizando sus aplicaciones monolíticas existentes por las siguientes razones:

- **Agilidad en la entrega, gracias al despliegue mejorado.** Los contenedores ofrecen un contrato consistente de despliegue entre desarrollo y operaciones. Cuando se usan contenedores, no se escuchará a los desarrolladores decir: "Funciona en mi máquina, ¿por qué no en producción?". Simplemente pueden decir: "Se ejecuta como un contenedor, por lo que se ejecutará en producción". La aplicación empaquetada, con todas sus dependencias, se puede ejecutar en cualquier entorno soportado basado en contenedor. Se ejecutará de la manera en que se pretendía ejecutar en todos los destinos de despliegue (desarrollo, control de calidad, *staging*, producción). Los contenedores eliminan la mayoría de las fricciones cuando se mueven de una fase a la siguiente, lo que mejora en gran medida el despliegue, y las aplicaciones pueden lanzarse más rápido.
- **Reducción de costes.** Los contenedores reducen los costes, ya sea por la consolidación y eliminación de hardware existente o por la ejecución de aplicaciones a una mayor densidad por unidad de hardware.
- **Portabilidad.** Los contenedores son modulares y portátiles. Los contenedores Docker se admiten en cualquier sistema operativo de servidor (Linux y Windows), en cualquier nube pública importante (Microsoft Azure, Amazon AWS, Google, IBM), y en entornos de nube local, privada o híbrida.
- **Control.** Los contenedores ofrecen un entorno flexible y seguro que se controla a nivel de contenedor. Un contenedor puede securizarse, aislarlo e incluso limitarse estableciendo políticas de restricción de ejecución en el contenedor. Como se detalla en la sección sobre Contenedores Windows, los Contenedores Windows Server 2016 y Hyper-V ofrecen opciones adicionales de soporte empresarial.

Las mejoras significativas en cuanto a agilidad, portabilidad y control llevan finalmente a reducciones de costes significativas cuando se usan contenedores para desarrollar y mantener aplicaciones.

¿Qué es Docker?

Docker es un [proyecto de código abierto](#) que automatiza el despliegue de aplicaciones como contenedores portátiles y autosuficientes, que se pueden ejecutar en la nube u *on-premises*. Docker también es la [compañía](#) que promueve y desarrolla esta tecnología. La compañía trabaja en colaboración con proveedores de nube, Linux y Windows, incluido Microsoft.

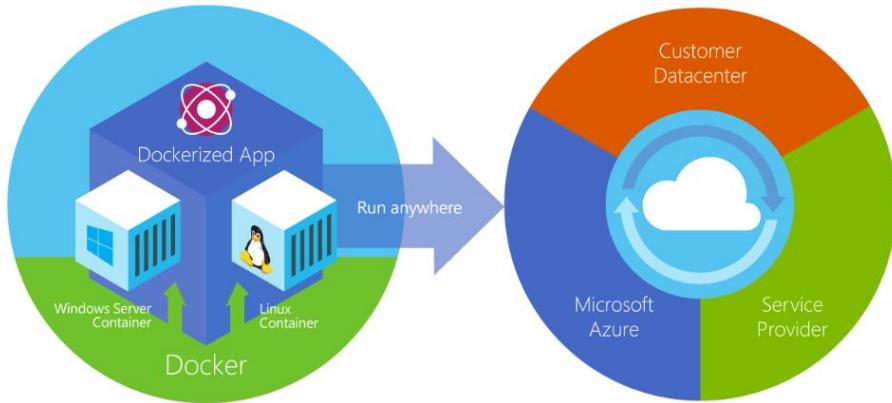


Figura 4-6. Docker despliega contenedores en todas las capas de la nube híbrida

Para alguien familiarizado con las máquinas virtuales, los contenedores pueden parecer notablemente similares. Un contenedor ejecuta un sistema operativo, tiene un sistema de archivos y se puede acceder a través de red, al igual que un sistema físico o virtual. Sin embargo, la tecnología y los conceptos detrás de los contenedores son muy diferentes a los de las máquinas virtuales. Desde el punto de vista del desarrollador, un contenedor debe tratarse más como un proceso único. De hecho, un contenedor tiene un solo punto de entrada para un proceso.

Los contenedores Docker (por simplicidad, contenedores) pueden ejecutarse de forma nativa en Linux y Windows. Cuando se ejecutan contenedores estándar, los Contenedores Windows solo se pueden ejecutar en *hosts* de Windows (un servidor *host* o una VM) y los contenedores Linux solo se pueden ejecutar en *hosts* Linux. Sin embargo, en las versiones recientes de los Contenedores Windows Server e Hyper-V, un contenedor de Linux también puede ejecutarse de forma nativa en Windows Server utilizando la tecnología de aislamiento de Hyper-V, que actualmente solo está disponible en Contenedores Windows Server.

En un futuro cercano, los entornos mixtos que tengan tanto contenedores Linux como Windows serán posibles e incluso comunes.

Beneficios de los Contenedores Windows para aplicaciones .NET existentes

Los beneficios del uso de Contenedores Windows son fundamentalmente los mismos que se obtienen de los contenedores en general. Usar Contenedores Windows es mejorar enormemente en la agilidad, la portabilidad y el control.

Cuando hablamos de aplicaciones .NET existentes, nos referimos principalmente a las aplicaciones tradicionales que se crearon mediante .NET Framework. Por ejemplo, pueden ser aplicaciones web ASP.NET tradicionales—no usan .NET Core, que es más nuevo y se ejecuta en plataformas Linux, Windows y MacOS.

La principal dependencia en .NET Framework es Windows. También tiene dependencias secundarias, como IIS, y System.Web en ASP.NET tradicional.

Una aplicación .NET Framework debe ejecutarse en Windows, punto. Si se desean contenerizar las aplicaciones .NET Framework existentes y no se puede o no se desea invertir en una migración a .NET Core ("Si funciona correctamente, no la migres"), la única opción que se tiene para usar contenedores es usar los Contenedores Windows.

Por lo tanto, uno de los principales beneficios de los Contenedores Windows es que ofrecen una forma de modernizar aplicaciones .NET Framework existentes que se ejecutan en Windows—a través de la contenerización. En definitiva, los Contenedores Windows ofrecen los beneficios que se buscan en general mediante el uso de contenedores: agilidad, portabilidad y mejor control.

Elegir un Sistema Operativo al cual orientarse con contenedores basados en .NET

Dada la diversidad de sistemas operativos que son compatibles con Docker, así como las diferencias entre .NET Framework y .NET Core, debe elegirse un sistema operativo específico, y sus versiones concretas, de acuerdo con en el *framework* que se esté utilizando.

Para Windows, se puede usar Windows Server Core o Windows Nano Server. Estas versiones de Windows proporcionan diferentes características (como IIS versus a un servidor web alojado en sí mismo, como Kestrel) que podrían necesitar las aplicaciones .NET Framework o .NET Core.

Para Linux, hay múltiples distribuciones disponibles y compatibles con las imágenes oficiales de Docker .NET (como Debian).

La Figura 4-7 muestra las versiones del sistema operativo a las que poder orientarse, según la versión de la aplicación de .NET Framework.

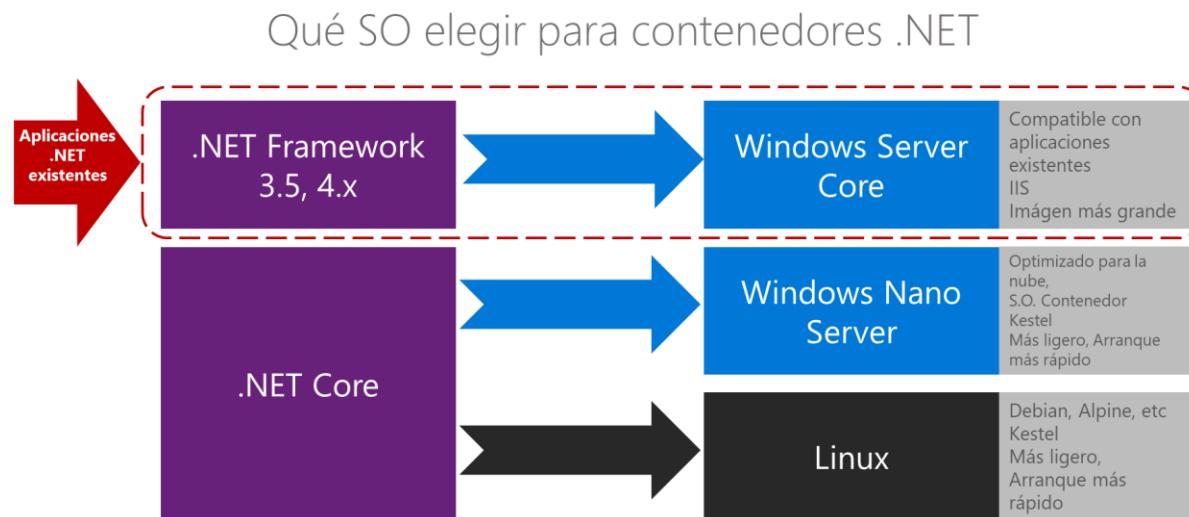


Figura 4-7. Sistemas operativos que elegir en base a la versión de .NET Framework

En escenarios de migración para aplicaciones existentes o *legacy* que se basan en aplicaciones de .NET Framework, las dependencias principales están en Windows e IIS. La única opción es usar imágenes de Docker basadas en Windows Server Core y .NET Framework.

Cuando se agrega el nombre de la imagen a un archivo Dockerfile, se puede seleccionar el sistema operativo y la versión con una etiqueta, como en los ejemplos siguientes para las imágenes de contenedor de Windows basadas en .NET Framework:

Etiqueta	Sistema y versión
microsoft/dotnet-framework:4.x-windowsservercore	.NET Framework 4.x en Windows Server Core
microsoft/aspnet:4.x-windowsservercore	.NET Framework 4.x con personalización ASP.NET adicional, en Windows Server Core

Para .NET Core (multiplataforma para Linux y Windows), las etiquetas se verían así:

Etiqueta	Sistema y versión
microsoft/dotnet:2.0.0-runtime	.NET Core 2.0 solo <i>runtime</i> , en Linux
microsoft/dotnet:2.0.0-runtime-nanoserver	.NET Core 2.0 solo <i>runtime</i> , en Windows Nano Server

Imágenes Multi-arch

A partir de mediados de 2017, también se puede usar una nueva función en Docker llamada imágenes [multi-arch](#). Las imágenes .NET Core Docker pueden usar etiquetas multi-arch. Los archivos Dockerfile ya no necesitan definir el sistema operativo al que están dirigidos. La función multi-arch permite que se use una sola etiqueta en múltiples configuraciones de máquina. Por ejemplo, con multi-arch, se puede usar una etiqueta común: **microsoft/dotnet:2.0.0-runtime**. Si se extrae esa etiqueta de un entorno contenedor de Linux, se obtendrá la imagen basada en Debian. Si se extrae esa etiqueta de un entorno de contenedor de Windows, se obtendrá la imagen basada en Nano Server.

Para las imágenes de .NET Framework, debido a que el .NET Framework tradicional solo admite Windows, no se puede usar la función multi-arch.

Tipos de Contenedores Windows

Al igual que los contenedores de Linux, los Contenedores Windows Server se administran mediante Docker Engine. A diferencia de los contenedores de Linux, los Contenedores Windows incluyen dos tipos diferentes de contenedores o *runtimes*—Contenedores Windows Server y aislamiento Hyper-V.

Contenedores Windows Server: Proporciona aislamiento de aplicaciones a través de la tecnología de aislamiento de proceso y *namespace*. Un contenedor de Windows Server comparte un *kernel* con el *host* contenedor y todos los contenedores que se ejecutan en el *host*. Estos contenedores no proporcionan un límite de seguridad hostil y no deberían utilizarse para aislar el código que no es de confianza. Debido al espacio compartido del *kernel*, estos contenedores requieren la misma versión y configuración del *kernel*.

Aislamiento Hyper-V: Expande el aislamiento proporcionado por los Contenedores Windows Server al ejecutar cada contenedor en una máquina virtual altamente optimizada. En esta configuración, el *kernel* del *host* del contenedor no se comparte con otros contenedores en el mismo *host*. Estos contenedores están diseñados para alojamiento hostil multiusuario, con las mismas garantías de seguridad que una máquina virtual. Debido a que estos contenedores no comparten el *kernel* con el *host* u otros contenedores en el *host*, pueden ejecutar *kernels* con diferentes versiones y configuraciones (con versiones soportadas). Por ejemplo, todos los Contenedores Windows en Windows 10 usan aislamiento de Hyper-V para utilizar la versión y la configuración del *kernel* de Windows Server.

La ejecución de un contenedor en Windows con o sin aislamiento de Hyper-V es una decisión en *runtime*. Se puede optar por crear el contenedor con aislamiento de Hyper-V inicialmente y, en *runtime*, ejecutarlo en cambio como un contenedor de Windows Server.

Recursos adicionales

- **Documentación sobre Contenedores Windows**
<https://docs.microsoft.com/virtualization/windowscontainers/>
- **Conceptos básicos sobre Contenedores Windows**
<https://docs.microsoft.com/virtualization/windowscontainers/about/>
- **Infographic: Microsoft and containers**
<https://info.microsoft.com/rs/157-GQE-382/images/Container%20infographic%201.4.17.pdf>

Cuando no desplegar a Contenedores Windows

Algunas tecnologías de Windows no son compatibles con los Contenedores Windows. En esos casos, es necesario migrar a VMs estándar, generalmente con solo Windows e IIS.

Casos no soportados en Contenedores Windows, a mediados de 2017:

- Microsoft Message Queuing (MSMQ) actualmente no está soportado en Contenedores Windows.
 - [UserVoice request forum](#)
 - [Discussion forum](#)
- Microsoft Distributed Transaction Coordinator (MSDTC) actualmente no está soportado en Contenedores Windows.
 - [GitHub issue](#)
- Microsoft Office actualmente no soporta contenedores.
 - [UserVoice request forum](#)

Para ver escenarios adicionales no soportados y hacer consultas a la comunidad, consultar el foro de UserVoice para Contenedores Windows:

<https://windowsserver.uservoice.com/forums/304624-containers>.

Recursos adicionales

- **Máquinas virtuales y contenedores en Azure**
<https://docs.microsoft.com/azure/virtual-machines/windows/containers>

Cuándo desplegar Contenedores Windows en infraestructura *on-premises* IaaS basada en VMs

Desplegar Contenedores Windows en infraestructura local (VM o servidores *bare-metal*) es una opción importante por varias razones:

- Es posible que la organización no esté lista para migrar a la nube o que simplemente no pueda moverse a la nube por un motivo comercial. Pero aún así, se pueden obtener los beneficios de usar Contenedores Windows en sus propios centros de datos.
- Es posible que tenga otros artefactos que se estén utilizando en *on-premises* y que puedan ralentizar el movimiento a la nube. Por ejemplo, dependencias de seguridad o autenticación con Windows Server Active Directory *on-premises* o cualquier otro activo *on-premises*.
- Si se comienzan a utilizar Contenedores Windows hoy, se puede hacer una migración por fases a la nube mañana desde una situación mucho mejor. Los Contenedores Windows se están convirtiendo en una unidad de despliegue para cualquier nube.

Cuándo desplegar Contenedores Windows a Azure VMs (nube IaaS)

Si la organización está utilizando máquinas virtuales Azure, incluso si también está utilizando Contenedores Windows, todavía se estará utilizando IaaS. Eso significa que se debe lidiar con las operaciones de infraestructura, los parches del sistema operativo de la máquina virtual y la complejidad de la infraestructura para aplicaciones altamente escalables cuando necesiten desplegar varias máquinas virtuales en una infraestructura de carga balanceada. Los principales escenarios para usar Contenedores Windows en una VM de Azure son:

- **Entorno de desarrollo/pruebas:** una VM en la nube es perfecta para el desarrollo y las pruebas en la nube. Se puede crear o detener rápidamente el entorno según las necesidades.
- **Necesidades de escalabilidad pequeñas y medianas:** en escenarios en los que se podrían necesitar solo un par de máquinas virtuales para el entorno de producción, administrar un número pequeño de máquinas virtuales podría ser asequible hasta que se pueda pasar a entornos de PaaS más avanzados, como los orquestadores.

- **Entorno de producción con herramientas de despliegue existentes:** Se puede pasar de un entorno *on-premises*, en el que se haya invertido en herramientas para realizar despliegues complejos, a VMs o servidores *bare-metal* (como Puppet o herramientas similares). Para pasar a la nube con cambios mínimos en los procedimientos de despliegue a entorno de producción, se pueden continuar usando esas herramientas para desplegar a máquinas virtuales Azure. Sin embargo, lo mejor será utilizar Contenedores Windows como la unidad de despliegue para mejorar la experiencia en el despliegue.

Cuando desplegar Contenedores Windows a Service Fabric

Las aplicaciones basadas en Contenedores Windows necesitarán rápidamente utilizar plataformas que se alejen aún más de VMs de IaaS. Esto es para mejorar la escalabilidad automatizada y la alta escalabilidad, y para obtener mejoras significativas en una experiencia de administración completa para despliegues, actualizaciones, versionado, *rollbacks* y monitorización del estado. Se pueden lograr estos objetivos con el orquestador Azure Service Fabric, disponible en la nube de Microsoft Azure, pero también *on-premises*, o incluso en otra nube.

Muchas organizaciones están moviendo directamente aplicaciones monolíticas existentes a contenedores por dos razones:

- Reducción de costes, ya sea debido a la consolidación y eliminación de hardware existente, o de ejecutar aplicaciones a densidad más alta.
- Un contrato de despliegue consistente entre desarrollo y operaciones.

Perseguir reducciones en los costes es comprensible, y es probable que todas las organizaciones persigan este objetivo. Un despliegue consistente es más difícil de evaluar, pero es igualmente importante. Un contrato consistente de despliegue incluye que los desarrolladores sean libres de elegir la tecnología a usar que les convenga, y el equipo de operaciones tenga una forma única de desplegar y administrar aplicaciones. Este acuerdo alivia el miedo de tener que lidiar con la complejidad de muchas tecnologías diferentes u obligar a los desarrolladores a trabajar solo con ciertas tecnologías. Básicamente, cada aplicación está contenida en una imagen de despliegue autónoma.

Algunas organizaciones continuarán modernizándose añadiendo microservicios (aplicaciones optimizadas para la nube y nativas para la nube). Muchas organizaciones se detendrán aquí (Preparada para DevOps en la Nube). Como se muestra en la Figura 4-8, estas organizaciones no se moverán a las arquitecturas de microservicios porque es posible que no las necesiten. En cualquier caso, ya obtienen los beneficios que proporcionan el uso de contenedores y Service Fabric: una experiencia de administración completa que incluye despliegue, actualizaciones, control de versiones, *rollbacks* y monitorización del estado.

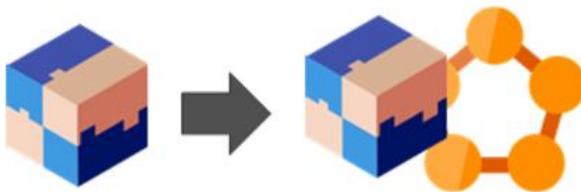


Figura 4-8. Migrar directamente una aplicación a Service Fabric

Un enfoque clave para Service Fabric es reutilizar el código existente y simplemente moverlo directamente. Por lo tanto, se pueden migrar aplicaciones actuales de .NET Framework, utilizando Contenedores Windows, y desplegarlas en Service Fabric. Será más fácil seguir modernizándose, eventualmente, agregando nuevos microservicios.

Al comparar Service Fabric con otros orquestadores, es importante destacar que Service Fabric está muy probado en la ejecución de aplicaciones y servicios basados en Windows. Service Fabric ha estado ejecutando aplicaciones y servicios basados en Windows, incluidos los productos Tier-1 de misión crítica de Microsoft, durante años. Fue el primer orquestador que tuvo disponibilidad general para Contenedores Windows (mayo de 2017). Otros contenedores, como Kubernetes, DC/OS y Docker Swarm, están más maduros en Linux, pero menos maduros que Service Fabric para aplicaciones basadas en Windows y Contenedores Windows.

El objetivo final de Service Fabric es reducir las complejidades de crear aplicaciones mediante el uso de un enfoque de microservicios. Esto es lo que eventualmente se querrá lograr para ciertos tipos de aplicaciones, para evitar rediseños caros. Se puede comenzar en pequeña escala cuando sea necesario, depreciar servicios, agregar nuevos servicios y evolucionar la aplicación según su uso por el cliente. Sabemos que hay muchos otros problemas que aún no se han resuelto para que los microservicios sean más accesibles para la mayoría de los desarrolladores. Si actualmente solo se está moviendo de manera directa una aplicación con Contenedores Windows, pero se está pensando en agregar microservicios basados en contenedores en el futuro, ese es el punto óptimo para el uso de Service Fabric.

Cuándo desplegar Contenedores Windows a Azure Container Service (p.e., Kubernetes)

Azure Container Service optimiza específicamente para Azure la configuración de herramientas y tecnologías populares de código abierto. Obteniendo una solución abierta que ofrece portabilidad tanto para los contenedores como para la configuración de la aplicación. Se puede elegir el tamaño, la cantidad de *hosts* y las herramientas del orquestador. Azure Container Service maneja la infraestructura por ti.

Si ya se está trabajando con orquestadores de código abierto como Kubernetes, Docker Swarm o DC/OS, no se necesitan cambiar las prácticas de administración existentes para mover las cargas de trabajo del contenedor a la nube. Se deben utilizar las herramientas de administración de aplicaciones con las que ya se esté familiarizado y conectarse a través de *endpoints* API estándar para el orquestador elegido.

Todos estos orquestadores suponen entornos maduros si se utilizan contenedores Docker de Linux, pero también son compatibles con Contenedores Windows desde 2017 (algunos anteriormente, otros más recientemente, según el orquestador).

Por ejemplo, en Kubernetes el soporte para contenedores es nativo, por lo que usar Contenedores Windows en Kubernetes también es muy efectivo y fiable (en *preview* hasta otoño de 2017).

Construir servicios resilientes preparados para la nube: Asumir fallos transitorios en la nube

La resiliencia es la capacidad de recuperarse de fallos y continuar funcionando. La resiliencia no se trata de evitar fallos, sino de aceptar el hecho de que ocurrirán fallos, y luego responder a ellos de una manera que evite el tiempo de inactividad o la pérdida de datos. El objetivo de la resiliencia es devolver la aplicación a un estado completamente funcional después de un fallo.

Una aplicación está lista para la nube cuando, como mínimo, implementa un modelo de resiliencia basado en software, en lugar de un modelo basado en hardware. Una aplicación en la nube debe asumir los fallos parciales que sin duda ocurrirán. Es necesario diseñar o refactorizar parcialmente la aplicación si se desea lograr resistencia a los fallos parciales esperados. Debe ser diseñada para hacer frente a fallos parciales, como interrupciones transitorias de la red y nodos, o VMs que se cuelgan en la nube. Incluso los contenedores que se mueven a un nodo diferente dentro de un grupo de orquestadores pueden causar fallos cortos intermitentes dentro de una aplicación.

Tratando los fallos parciales

En una aplicación basada en la nube, existe un riesgo constante de fallo parcial. Por ejemplo, una sola instancia de un sitio web o un contenedor puede fallar, o puede no estar disponible o no responder por un periodo corto de tiempo. O bien, una sola máquina virtual o servidor podrían bloquearse.

Debido a que los clientes y los servicios son procesos separados, es posible que un servicio no pueda responder de manera oportuna a la solicitud de un cliente. Es posible que el servicio esté sobrecargado y responda muy lentamente a las solicitudes, o simplemente podría no ser accesible durante un breve período de tiempo debido a problemas de red.

Por ejemplo, considérese una aplicación .NET monolítica que accede a una base de datos en Azure SQL Database. Si la base de datos SQL de Azure o cualquier otro servicio de terceros no responde durante un breve período de tiempo (una base de datos SQL de Azure puede moverse a un nodo o servidor diferente y no responder durante unos segundos), cuando el usuario intente realizar alguna acción, la aplicación puede bloquearse y mostrar una excepción en ese momento exacto.

Un escenario similar podría ocurrir en una aplicación que consume servicios HTTP. Es posible que la red o el servicio en sí no estén disponibles en la nube durante un fallo breve y transitorio.

Una aplicación resiliente como la que se muestra en la Figura 4-9 debería implementar técnicas como "reintentos con demora exponencial" para dar a la aplicación la oportunidad de manejar los fallos transitorios en recursos. También se deben usar "interruptores automáticos" en las aplicaciones. Un interruptor de circuito impide que una aplicación intente acceder a un recurso cuando en realidad

existe una situación de fallo a largo plazo. Al usar un interruptor de circuito, la aplicación evita provocar una denegación de servicio a sí mismo.

Comunicación resiliente en aplicaciones en la nube

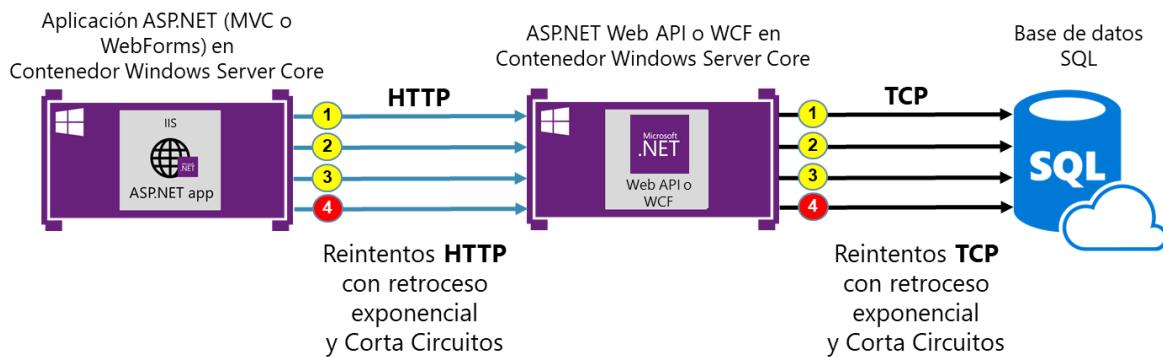


Figura 4-9. Fallos parciales manejados por reintentos con demora exponencial

Se pueden utilizar estas técnicas tanto en recursos HTTP como en recursos de bases de datos. En la Figura 4-9, la aplicación se basa en una arquitectura de 3 niveles, por lo que necesita estas técnicas en el nivel de servicios (HTTP) y en el nivel de datos (TCP). En una aplicación monolítica que utiliza solo un nivel de aplicación único además de la base de datos (sin servicios adicionales o microservicios), puede ser suficiente con el manejo de fallos transitorios en el nivel de conexión de la base de datos. En ese escenario, generalmente solo se requiere una configuración particular de la conexión a la base de datos.

Al implementar comunicaciones flexibles que accedan a la base de datos, dependiendo de la versión de .NET que se esté utilizando, puede ser muy sencillo (por ejemplo, [con Entity Framework 6 o posterior](#), solo se trata de configurar la conexión de la base de datos). O bien, es posible que se necesiten utilizar bibliotecas adicionales como el [Transient Fault Handling Application Block](#) (para versiones anteriores de .NET) o incluso implementar una biblioteca propia.

Cuando se implementan reintentos HTTP y corta circuitos, la recomendación para .NET es usar la biblioteca [Polly](#), que se enfoca a .NET 4.0, .NET 4.5 y .NET Standard 1.1, y que incluye compatibilidad con .NET Core.

Para aprender a implementar estrategias para manejar fallos parciales en la nube, consultar las siguientes referencias.

Recursos adicionales

- **Implementación de comunicación resiliente para manejar fallos parciales**
<https://docs.microsoft.com/dotnet/standard/microservices-architecture/implement-resilient-applications/partial-failure-strategies>
- **Resiliencia de conexión en Entity Framework y lógica de reinicio (versión 6 y posterior)**
[https://msdn.microsoft.com/library/dn456835\(v=vs.113\).aspx](https://msdn.microsoft.com/library/dn456835(v=vs.113).aspx)
- **El libro de manejo de fallos transitorios en aplicaciones**

[https://msdn.microsoft.com/library/hh680934\(v=pandp.50\).aspx](https://msdn.microsoft.com/library/hh680934(v=pandp.50).aspx)

- **Librería Polly para comunicación HTTP resiliente**
<https://github.com/App-vNext/Polly>

Modernizar aplicaciones con monitorización y telemetría

Cuando se ejecuta una aplicación en producción, es fundamental que se tenga información sobre su rendimiento. ¿Está funcionando a alto nivel? ¿Los usuarios reciben errores o la aplicación es estable y fiable? Se necesita de una supervisión exhaustiva del rendimiento, alertas potentes y cuadros de mando para ayudar a garantizar que la aplicación esté disponible y tenga el rendimiento esperado. También se debe ser capaz de ver rápidamente si hay un problema, determinar cuántos clientes se ven afectados y realizar un análisis de la causa raíz para encontrar y solucionar el problema.

Monitorización de aplicaciones con Application Insights

Application Insights es un servicio extensible de Application Performance Management (APM) para desarrolladores web que trabajan en múltiples plataformas. Se utiliza para monitorizar una aplicación web en vivo. Application Insights detecta automáticamente anomalías de rendimiento. Incluye poderosas herramientas de análisis para ayudar a diagnosticar problemas y a comprender lo que los usuarios realmente hacen con la aplicación. Application Insights está diseñado para ayudar a mejorar continuamente el rendimiento y la usabilidad. Funciona para aplicaciones en una amplia variedad de plataformas, incluidas .NET, Node.js y J2EE, ya sean *on-premises* o en la nube. Application Insights se integra con procesos DevOps y tiene puntos de conexión a una amplia variedad de herramientas de desarrollo.

La Figura 4-10 muestra un ejemplo de cómo Application Insights supervisa una aplicación y cómo hace llegar la información a un panel.

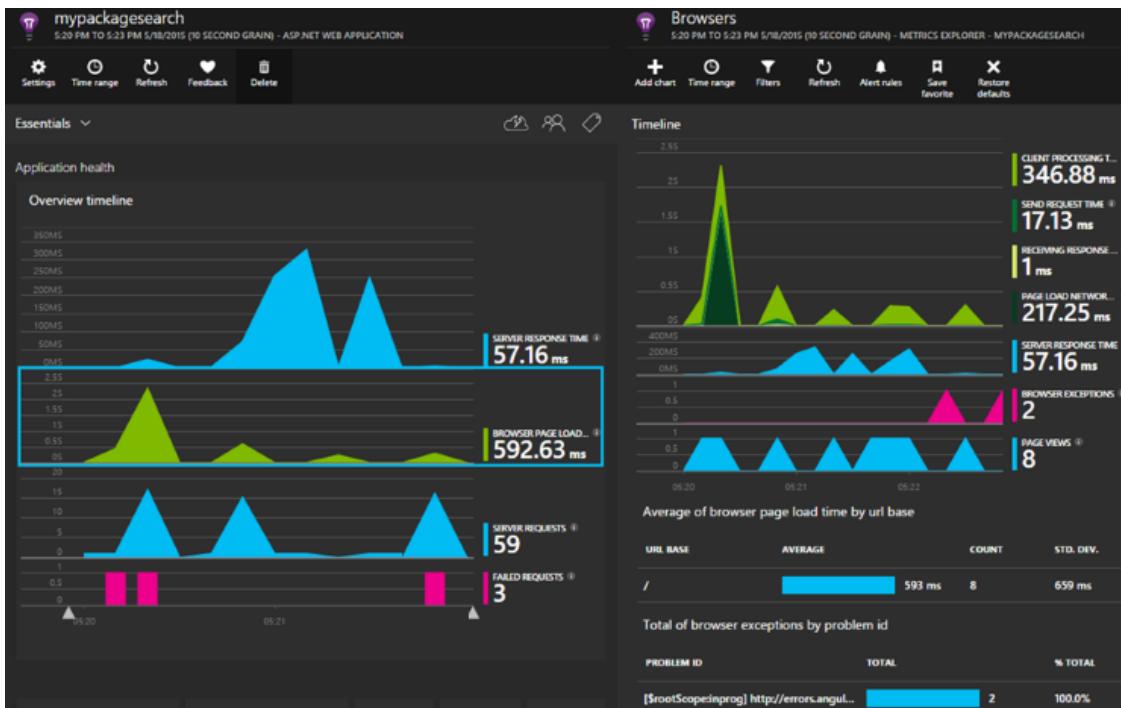


Figure 4-10. Application Insights monitoring dashboard

Monitorizar infraestructura Docker con Log Analytics y su solución de monitorización de Contenedor

[Azure Log Analytics](#) es parte de la solución de monitorización general de Microsoft Azure. También es un servicio en [Operations Management Suite \(OMS\)](#). Log Analytics supervisa los entornos en la nube y *on-premises* (OMS para *on-premises*) para ayudar a mantener la disponibilidad y el rendimiento. Recopila datos generados por recursos en la nube y *on-premises*, y desde otras herramientas de monitorización, para proporcionar análisis a través de múltiples fuentes.

En relación con los *logs* de infraestructura de Azure, Log Analytics, como servicio de Azure, ingiere datos de *logs* y métricas de otros servicios de Azure (a través de Azure Monitor), máquinas virtuales Azure, contenedores Docker, y *on-premises* u otras infraestructuras en la nube. Log Analytics ofrece búsqueda flexible en el *log* y análisis, listos para utilizar sobre los datos. Proporciona herramientas completas que se pueden usar para analizar datos en todas las fuentes, permite consultas complejas en todos los *logs* y puede alertar proactivamente en función de condiciones específicas. Incluso puede recopilar datos personalizados en el repositorio central de Log Analytics, donde pueden consultarse y visualizarse. También se pueden aprovechar las soluciones integradas de Log Analytics para obtener de inmediato información sobre la seguridad y la funcionalidad de la infraestructura.

Es posible acceder a Log Analytics a través del portal de OMS o del portal de Azure, se ejecuta en cualquier navegador, y proporciona acceso a las configuraciones y múltiples herramientas para analizar y actuar sobre los datos recopilados.

La solución de [Supervisión de Contenedores](#) en Log Analytics ayuda a ver y administrar los servidores Docker y Contenedores Windows desde una sola ubicación. La solución muestra qué contenedores se están ejecutando, qué imagen de contenedor están ejecutando y dónde se están ejecutando los contenedores. Se puede ver información detallada de auditoría, incluidos los comandos que se utilizan con los contenedores. También se pueden solucionar problemas relacionados con los contenedores mediante la visualización y búsqueda de *logs* centralizados, sin necesidad de ver remotamente los Docker o Windows *hosts*. Se pueden encontrar contenedores que puedan ser “ruidosos” y que consuman excesivos recursos en un *host*. Además, se puede ver el uso centralizado de la CPU, la memoria, el almacenamiento y la red, y la información de rendimiento, para los contenedores. En los ordenadores que ejecuten Windows, se pueden centralizar y comparar logs de los contenedores Windows Server, Hyper-V y Docker. La solución es compatible con los siguientes orquestadores de contenedores:

- Docker Swarm
- DC/OS
- Kubernetes
- Service Fabric
- Red Hat OpenShift

La Figura 4-11 muestra las relaciones entre varios *hosts* contenedores, agentes y OMS.

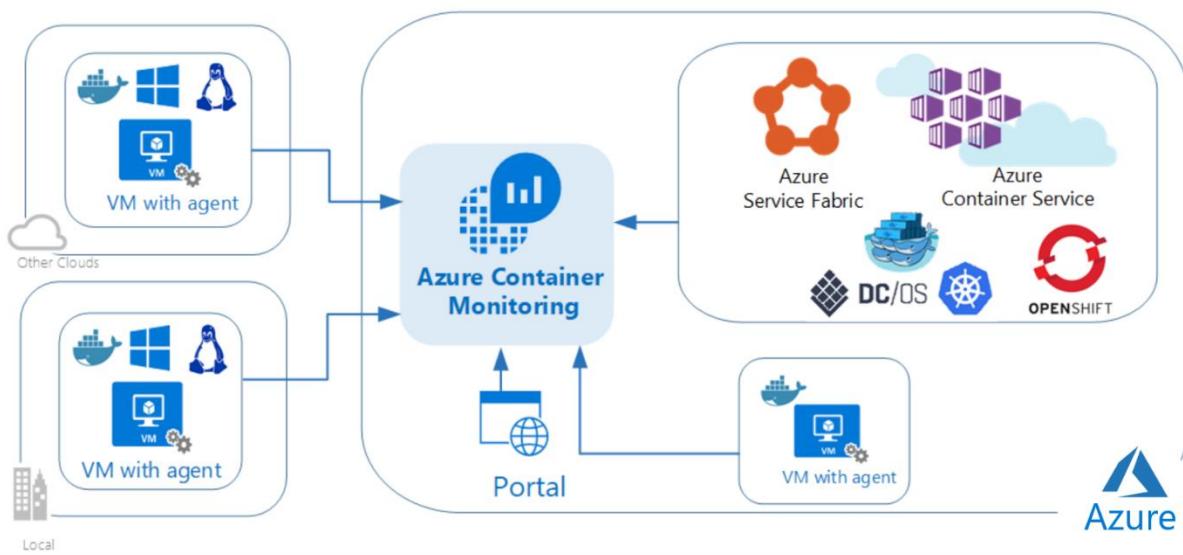


Figura 4-11. Solución con Log Analytics Container Monitoring

Se puede usar la solución Log Analytics Container Monitoring para:

- Ver información sobre todos los *hosts* contenedores en una sola ubicación.
- Saber qué contenedores se están ejecutando, qué imagen están ejecutando y dónde se están ejecutando.
- Ver un *log* de auditoría de acciones en contenedores.

- Solucionar problemas mediante la visualización y búsqueda de *logs* centralizados sin necesidad de iniciar sesión remota en los *hosts Docker*.
- Buscar contenedores que puedan ser "vecinos ruidosos" y que consuman en exceso recursos de un *host*.
- Ver el uso centralizado de la CPU, la memoria, el almacenamiento y la red, y la información de rendimiento, para los contenedores.

Recursos adicionales

- **Visión general de supervisión en Microsoft Azure**
<https://docs.microsoft.com/azure/monitoring-and-diagnostics/monitoring-overview>
- **¿Qué es Application Insights?**
<https://docs.microsoft.com/azure/application-insights/app-insights-overview>
- **¿Qué es Log Analytics?**
<https://docs.microsoft.com/azure/log-analytics/log-analytics-overview>
- **Supervisión de Contenedores en Log Analytics**
<https://docs.microsoft.com/azure/log-analytics/log-analytics-containers>
- **Visión general de Azure Monitor**
<https://docs.microsoft.com/azure/monitoring-and-diagnostics/monitoring-overview-azure-monitor>
- **¿Qué es Operations Management Suite (OMS)?**
<https://docs.microsoft.com/azure/operations-management-suite/operations-management-suite-overview>
- **Supervisión de Contenedores Windows Server en Service Fabric con OMS**
<https://docs.microsoft.com/azure/service-fabric/service-fabric-diagnostics-containers-windowsserver>

Modernizar el ciclo de vida de las aplicaciones, con procesos CI/CD y herramientas DevOps en la nube

Las empresas de hoy necesitan innovar a un ritmo rápido para ser competitivas en el mercado. Ofrecer aplicaciones modernas de alta calidad requiere de herramientas y procesos DevOps que son fundamentales para implementar este ciclo constante de innovación. Con las herramientas DevOps correctas, los desarrolladores pueden simplificar el despliegue continuo y llevar aplicaciones innovadoras a manos de los usuarios de forma más rápida.

Aunque las prácticas de integración y despliegue continuas están bien establecidas, la introducción de contenedores añade nuevas consideraciones, particularmente cuando se trabaja con aplicaciones de contenedores múltiples.

Visual Studio Team Services admite la integración y despliegue continuo de aplicaciones multi-contenedores a una variedad de entornos, a través de las tareas oficiales de despliegue de Team Services:

- [Desplegar a una VM independiente de Docker Host](#) (Linux o Windows Server 2016 o posterior)
- [Desplegar a Service Fabric](#)
- [Desplegar a Azure Container Service – Kubernetes](#)

Pero también se puede desplegar a [Docker Swarm](#) o DC/OS mediante tareas basadas en secuencias de comandos de Team Services.

Para continuar facilitando la agilidad en el despliegue, estas herramientas brindan experiencias de despliegue excelentes de tipo desarrollo-a-pruebas-a-producción para cargas de trabajo de contenedores, con varias opciones de soluciones para desarrollo y CI/CD.

La Figura 4-12 muestra un proceso de despliegue continuo a un clúster de Kubernetes en Azure Container Service.

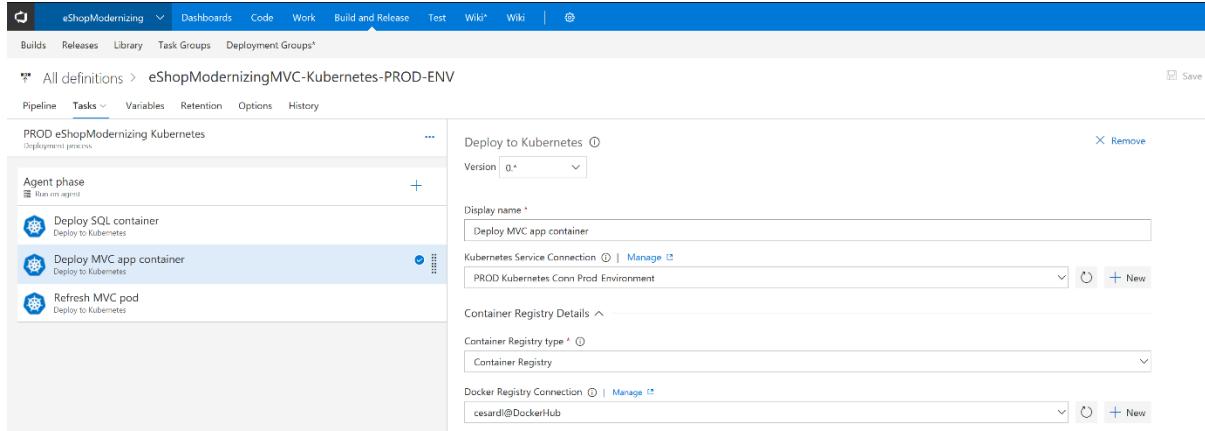


Figura 4-12. Proceso de despliegue continuo en Visual Studio Team Services, desplegando a un cluster Kubernetes

Migrar a escenarios híbridos en la nube

Algunas organizaciones y empresas no pueden migrar algunas de sus aplicaciones a nubes públicas como Microsoft Azure o cualquier otra nube pública debido a regulaciones o a sus propias políticas. Sin embargo, es probable que cualquier organización se beneficie de tener algunas de sus aplicaciones en la nube pública y otras aplicaciones *on-premises*. Sin embargo, un entorno mixto puede generar una complejidad excesiva en los entornos debido a las diferentes plataformas y tecnologías utilizadas en las nubes públicas frente a los entornos *on-premises*.

Microsoft proporciona la mejor solución de nube híbrida, una en la que se pueden optimizar tanto los activos *on-premises* como en la nube pública, mientras garantiza la coherencia en una nube híbrida de Azure. Se pueden maximizar las habilidades existentes, y obtener un enfoque flexible y unificado, para crear aplicaciones que se puedan ejecutar en la nube u *on-premises*, gracias a Azure Stack (*on-premises*) y Azure (nube pública).

Cuando se trata de seguridad, se pueden centralizar la administración y la seguridad en la nube híbrida. Se puede obtener el control de todos los activos, desde el centro de datos a la nube, al proporcionar un inicio de sesión único tanto a las aplicaciones locales como en la nube. Esto se logra al extender Active Directory a una nube híbrida y al usar la administración de identidades.

Finalmente, se pueden distribuir y analizar datos sin problemas, usar los mismos lenguajes de consulta para la nube y los activos *on-premises*, y aplicar análisis y aprendizaje profundo (Deep Learning) en Azure para enriquecer los datos, independientemente de su origen.

Azure Stack

Azure Stack es una plataforma híbrida en la nube que permite entregar servicios de Azure desde el centro de datos de la organización. Azure Stack está diseñado para admitir nuevas opciones para las aplicaciones modernas en escenarios clave, como entornos extremos y no conectados, o para cumplir con requisitos específicos de seguridad y cumplimiento.

La Figura 4-13 muestra una descripción general de la verdadera plataforma de nube híbrida que ofrece Microsoft.

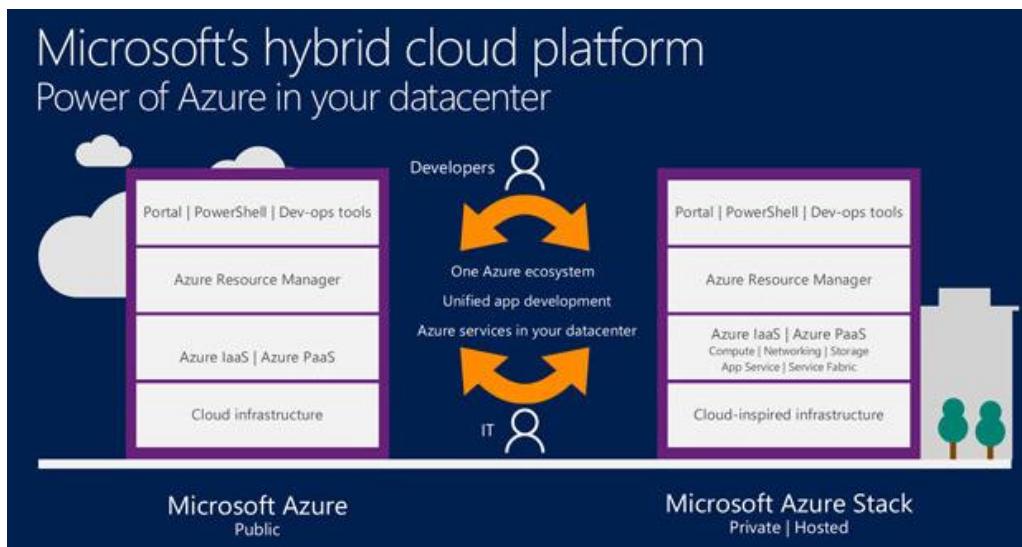


Figura 4-13. Plataforma de nube híbrida con Azure Stack y Azure

Azure Stack se ofrece en dos opciones de despliegue, para cubrir cada necesidad:

- Sistemas integrados de Azure Stack
- Azure Stack Development Kit

Sistemas integrados de Azure Stack

Los sistemas integrados de Azure Stack se ofrecen a través de una asociación de Microsoft y otros socios de hardware. La asociación crea una solución equilibrada que ofrece innovación basada en la nube y simplicidad en la administración. Debido a que Azure Stack se ofrece como un sistema integrado de hardware y software, se obtiene la cantidad suficiente de flexibilidad y control, al mismo tiempo que se adopta innovación desde la nube. Los sistemas integrados de Azure Stack varían en tamaño de 4 a 12 nodos, y son compatibles conjuntamente entre el socio del hardware y Microsoft. Utiliza los sistemas integrados Azure Stack para implementar nuevos escenarios para las cargas de trabajo en producción.

Azure Stack Development Kit

Microsoft Azure Stack Development Kit es una implementación de nodo único de Azure Stack, que se puede usar para evaluar y aprender sobre Azure Stack. También se puede usar Azure Stack Development Kit como un entorno de desarrollador, donde es posible desarrollar utilizando APIs y herramientas que sean consistentes con Azure. Azure Stack Development Kit no está destinado a ser utilizado como un entorno de producción.

Recursos adicionales

- **Nube híbrida de Azure**
<https://azure.microsoft.com/overview/hybrid-cloud/>
- **Azure Stack**
<https://azure.microsoft.com/overview/azure-stack/>
- **Active Directory Service Accounts para Contenedores Windows**

<https://docs.microsoft.com/virtualization/windowscontainers/manage-containers/manage-serviceaccounts>

- **Crear un contenedor con soporte a Active Directory**
<https://blogs.msdn.microsoft.com/containerstuff/2017/01/30/create-a-container-with-active-directory-support/>
- **Ventajas del licenciamiento Azure Hybrid**
<https://azure.microsoft.com/pricing/hybrid-use-benefit/>

Tutoriales y visión técnica inicial

Para limitar el tamaño de este *eBook*, hemos creado documentación técnica adicional y tutoriales paso a paso completos que están disponibles en un repositorio de GitHub. La serie de tutoriales en línea que se describen en este capítulo cubren la configuración paso a paso de los múltiples entornos basados en Contenedores Windows y su despliegue a Azure.

Las siguientes secciones explican de qué trata cada tutorial—sus objetivos, su visión a alto nivel—y proporcionan un diagrama de las tareas que están involucradas. Se pueden obtener los tutoriales en la wiki de las aplicaciones eShopModernizing en el repo GitHub <https://github.com/dotnet-architecture/eShopModernizing/wiki>.

Listado de tutoriales paso-a-paso técnicos

Los siguientes tutoriales de inicio proporcionan orientación técnica coherente y exhaustiva para las aplicaciones de ejemplo que se pueden migrar directamente mediante el uso de contenedores, y luego moverse utilizando múltiples opciones de despliegue en Azure.

Cada uno de los siguientes tutoriales utiliza las nuevas aplicaciones de ejemplo eShopLegacy y eShopModernizing, que están disponibles en GitHub en <https://github.com/dotnet-architecture/eShopModernizing>.

- **Tour de aplicaciones *legacy* en eShop**
- **Contenerizar aplicaciones .NET existentes con Contenedores Windows**
- **Desplegar una aplicación basada en Contenedores Windows a Azure VMs**
- **Desplegar aplicaciones basadas en Contenedores Windows a Kubernetes en Azure Container Service**
- **Desplegar aplicaciones basadas en Contenedores Windows a Azure Service Fabric**

Tutorial 1: Tour de aplicaciones *legacy* en eShop

Disponibilidad del tutorial técnico paso-a-paso

El tutorial técnico completo está disponible en la wiki del repo GitHub de eShopModernizing:

<https://github.com/dotnet-architecture/eShopModernizing/wiki/01.-Tour-on-eShopModernizing-apps-implementation-code>

Visión general

En este tutorial, se puede explorar la implementación inicial de dos aplicaciones *legacy* de ejemplo. Ambas aplicaciones de ejemplo tienen una arquitectura monolítica y fueron creadas mediante el uso de ASP.NET clásico. Una aplicación se basa en ASP.NET 4.x MVC; la segunda aplicación se basa en ASP.NET 4.x Web Forms. Ambas aplicaciones están en el [repo GitHub de eShopModernizing](#).

Se pueden contenerizar ambas aplicaciones de ejemplo, de forma similar a como puede contenerizar una aplicación clásica de [Windows Communication Foundation](#) (WCF) para consumir como una aplicación de escritorio. Para ver un ejemplo, consulte [eShopModernizingWCFWinForms](#).

Objetivos

El objetivo principal de este tutorial es simplemente familiarizarse con estas aplicaciones y con su código y configuración. Se pueden configurar las aplicaciones para que generen y usen datos simulados, sin usar la base de datos SQL, para fines de prueba. Esta configuración opcional se basa en la inyección de dependencia, en una forma desacoplada.

Escenario

La Figura 5-1 muestra el escenario simple de las aplicaciones *legacy* originales.

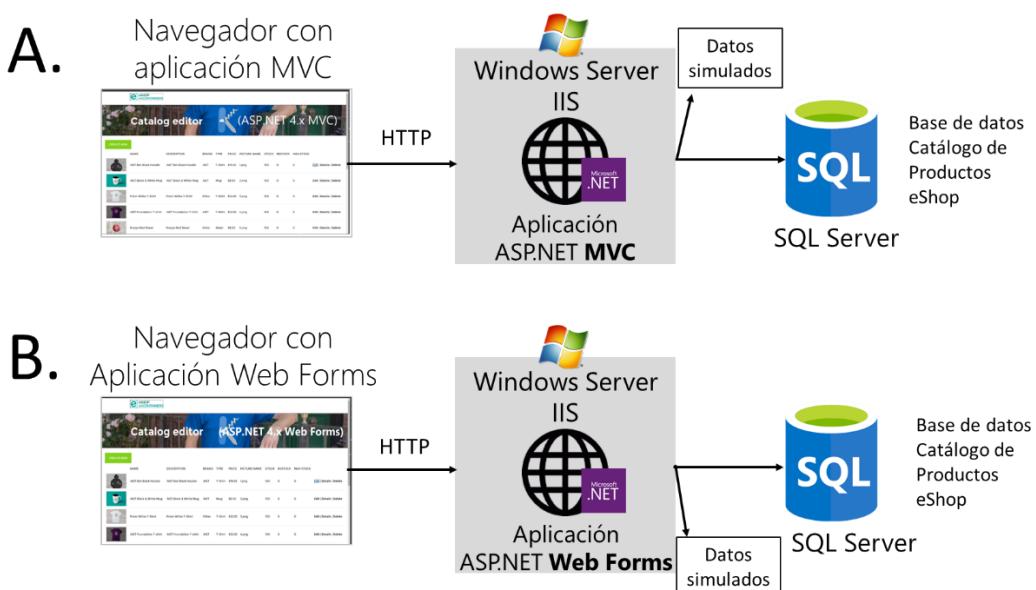


Figura 5-1. Escenario de arquitectura simple de aplicaciones legacy originales

Desde una perspectiva de negocio, ambas aplicaciones ofrecen las mismas funciones de gestión de catálogos. Los miembros del equipo empresarial de eShop usarían la aplicación para ver y editar el catálogo de productos. La Figura 5-2 muestra las capturas de pantalla iniciales de la aplicación.

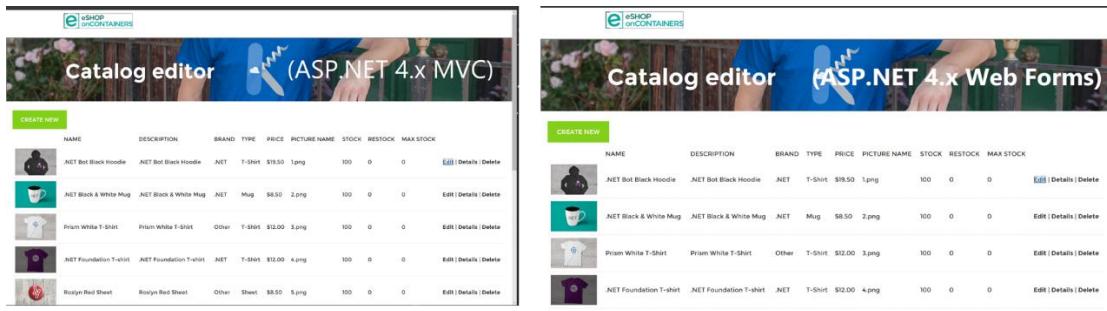


Figura 5-2. Aplicaciones de Web Forms de ASP.NET MVC y ASP.NET (existentes/tecnologías legacy)

Estas son aplicaciones web que se utilizan para explorar y modificar entradas de catálogo. El hecho de que ambas aplicaciones ofrezcan las mismas características de negocio/funcionales es simplemente para fines de comparación. Se puede ver un proceso de modernización similar para las aplicaciones que se crearon utilizando los frameworks ASP.NET MVC y ASP.NET Web Forms.

Las dependencias en ASP.NET 4.x o versiones anteriores (ya sea para MVC o para formularios web) significan que estas aplicaciones no se ejecutarán en .NET Core a menos que el código se reescriba completamente utilizando ASP.NET Core MVC. Esto demuestra el punto de que, si no se desea volver a diseñar o reescribir el código, se pueden contenerizar aplicaciones existentes y seguir usando las mismas tecnologías .NET y mismo código. Puede verse cómo es posible ejecutar aplicaciones como estas en contenedores, sin ningún cambio en el código *legacy*.

Beneficios

Los beneficios de este tutorial son simples: familiarícese con el código y la configuración de la aplicación, basada en la inyección de dependencia. Luego, experimentar con este enfoque cuando se contenerice y despliegue en múltiples entornos en el futuro.

Siguientes pasos

Explorar este contenido más a fondo en la wiki de GitHub:

<https://github.com/dotnet-architecture/eShopModernizing/wiki/01.-Tour-on-eShopModernizing-apps-implementation-code>

Tutorial 2: Contenerizar aplicaciones .NET existentes con Contenedores Windows

Disponibilidad del tutorial técnico paso-a-paso

El tutorial técnico completo está disponible en la wiki del repo GitHub de eShopModernizing:

<https://github.com/dotnet-architecture/eShopModernizing/wiki/02.-How-to-containerized-the-.NET-Framework-web-apps-with-Windows-Containers-and-Docker>

Visión general

Usar Contenedores Windows para mejorar el despliegue de aplicaciones .NET existentes, como las basadas en MVC, Web Forms o WCF, en entornos de producción, desarrollo y pruebas.

Objetivos

El objetivo de este tutorial es enseñar varias opciones para contener una aplicación existente de .NET Framework. Se puede:

- Contanerizar una aplicación utilizando [Visual Studio 2017 Tools for Docker](#) (Visual Studio 2017 o versiones posteriores).
- Contanerizar una aplicación añadiendo manualmente un [Dockerfile](#), y después utilizando el [Docker CLI](#).
- Contanerizar una aplicación utilizando la herramienta [Img2Docker](#) (una herramienta de código abierto de Docker).

Este tutorial se centra en el enfoque de Visual Studio 2017 Tools para Docker, pero los otros dos enfoques son bastante similares en cuanto al uso de Dockerfiles.

Escenario

La Figura 5-3 muestra el escenario para las aplicaciones *legacy* de eShop en contenedores.

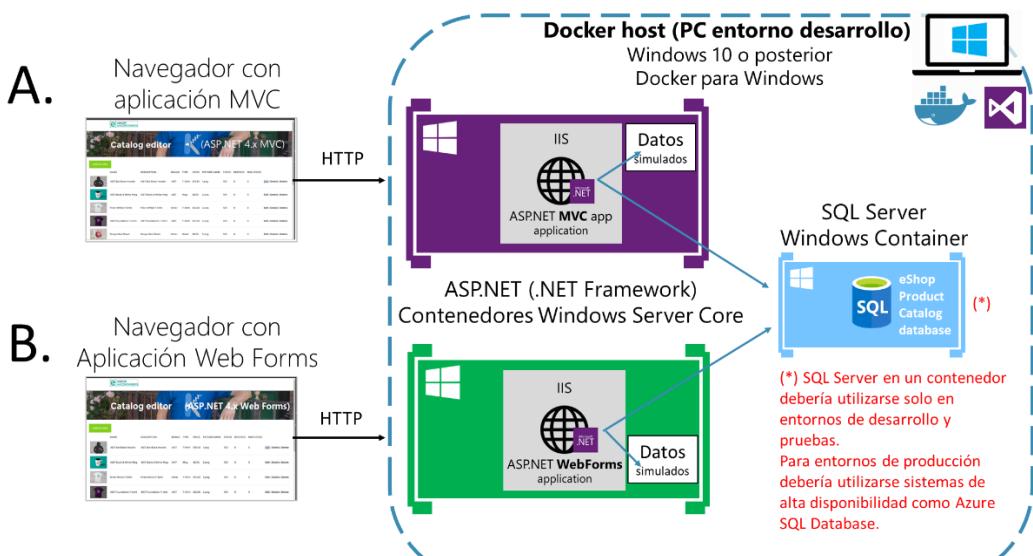


Figura 5-3. Diagrama de arquitectura simplificada de aplicaciones contenerizadas en un entorno de desarrollo

Beneficios

Existen ventajas al ejecutar una aplicación monolítica en un contenedor. Primero, se crea una imagen para la aplicación. A partir de ese momento, cada despliegue se ejecuta en el mismo entorno. Cada contenedor utiliza la misma versión de sistema operativo, tiene la misma versión de dependencias instalada, utiliza la misma versión de .NET Framework y se genera utilizando el mismo proceso. Básicamente, se controlan las dependencias de la aplicación mediante el uso de una imagen Docker. Las dependencias viajan con la aplicación cuando se despliegan los contenedores.

Un beneficio adicional es que los desarrolladores pueden ejecutar la aplicación en el entorno consistente proporcionado por Contenedores Windows. Los problemas que aparecen con ciertas versiones se pueden detectar de inmediato, en lugar de salir a la luz en un entorno de *staging* o producción. Las diferencias en los entornos de desarrollo utilizados por los miembros del equipo de desarrollo importan menos cuando las aplicaciones se ejecutan en contenedores.

Las aplicaciones en contenedores también tienen una curva de escalado más plana. Las aplicaciones en contenedores permiten tener más instancias de aplicaciones y servicios (basadas en contenedores) en una VM o máquina física en comparación con los despliegues habituales de aplicaciones por máquina. Esto se traduce en mayor densidad y menos recursos necesarios, especialmente cuando se utilizan orquestadores como Kubernetes o Service Fabric.

La contenerización, en situaciones ideales, no requiere realizar ningún cambio en el código de la aplicación (C #). En la mayoría de los escenarios, solo se necesitan los archivos de metadatos de despliegue de Docker (archivos Dockerfiles y Docker Compose).

Siguientes pasos

Explorar el siguiente contenido para mayor detalle en la wiki de GitHub:

<https://github.com/dotnet-architecture/eShopModernizing/wiki/02.-How-to-containerized-the-.NET-Framework-web-apps-with-Windows-Containers-and-Docker>

Tutorial 3: Desplegar una aplicación basada en Contenedores Windows a Azure VMs

Disponibilidad del tutorial técnico paso-a-paso

El tutorial técnico completo está disponible en la wiki del repo GitHub de eShopModernizing:

[https://github.com/dotnet-architecture/eShopModernizing/wiki/03.-How-to-deploy-your-Windows-Containers-based-app-into-Azure-VMs-\(Including-CI-CD\)](https://github.com/dotnet-architecture/eShopModernizing/wiki/03.-How-to-deploy-your-Windows-Containers-based-app-into-Azure-VMs-(Including-CI-CD))

Visión General

Desplegar a un *host Docker* en una VM de Windows Server 2016 en Azure permite configurar rápidamente entornos de desarrollo/pruebas/*staging*. También proporciona un lugar común, para los *testers* o usuarios de negocio, para validar la aplicación. Las VM también pueden ser entornos de producción IaaS válidos.

Objetivos

El objetivo de este tutorial es mostrar las múltiples alternativas que se tienen cuando se despliegan Contenedores Windows en máquinas virtuales Azure basadas en Windows Server 2016 o versiones posteriores.

Escenarios

Existen varios escenarios cubiertos en este tutorial.

Escenario A: Despliegue a una Azure VM desde un PC de desarrollo a través de una conexión Docker Engine

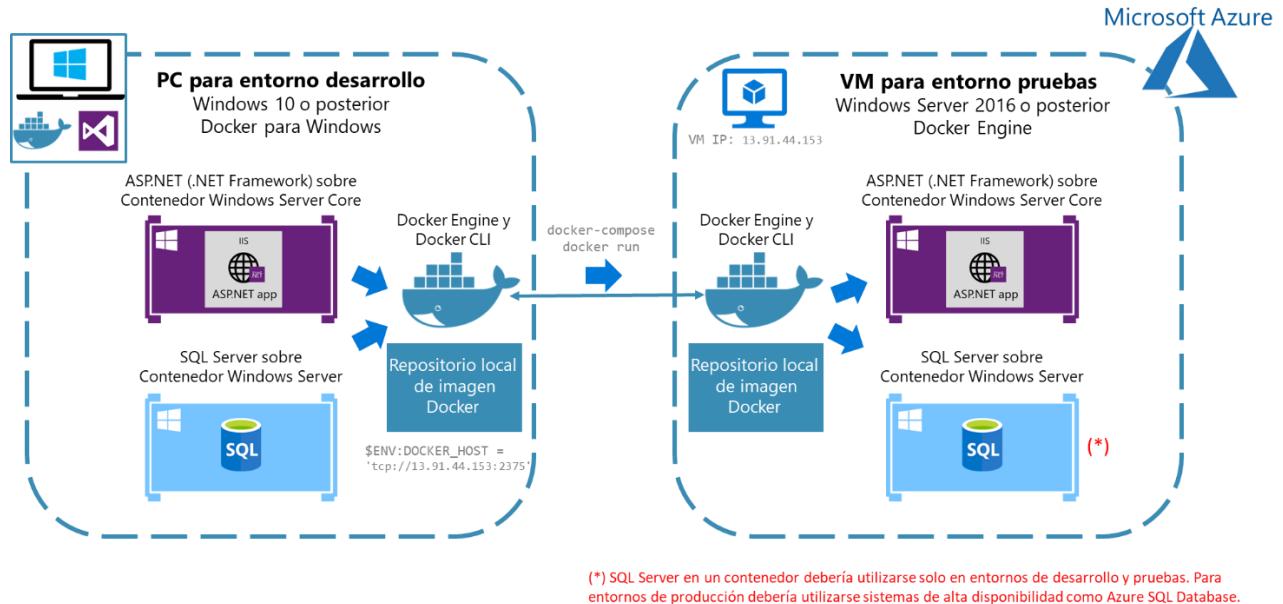


Figura 5-4. Despliegue a una Azure VM desde un PC de desarrollo a través de una conexión Docker Engine

Escenario B: Despliegue a una Azure VM a través de Docker Registry

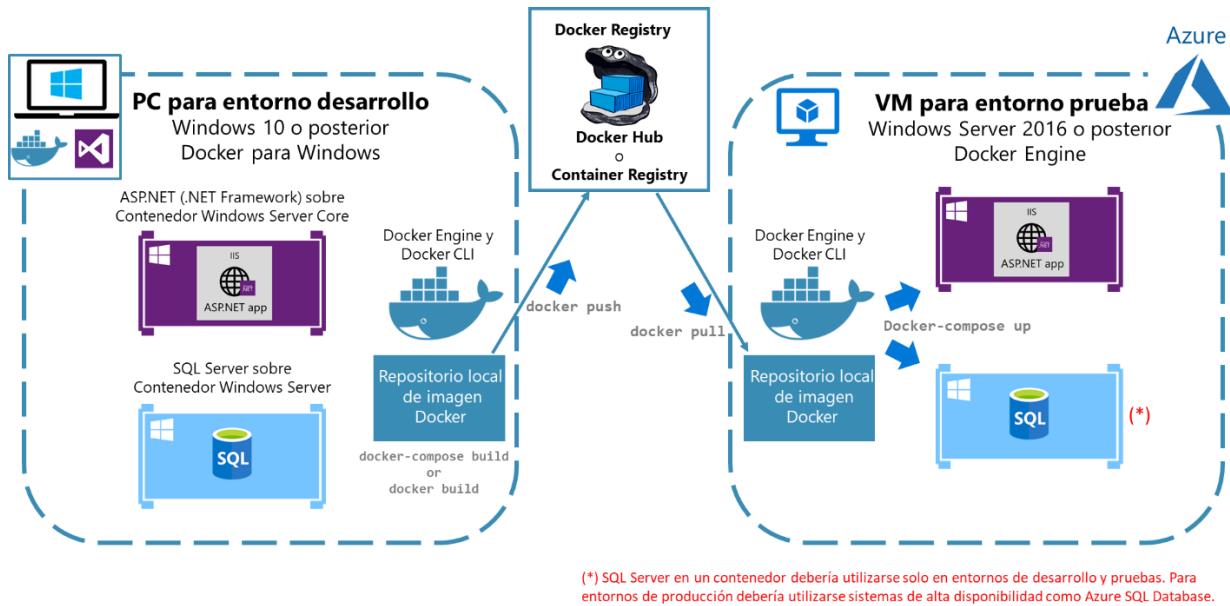


Figura 5-5. Despliegue a Azure VM a través de Docker Registry

Escenario C: Despliegue a una Azure VM desde procesos CI/CD en Visual Studio Team Services

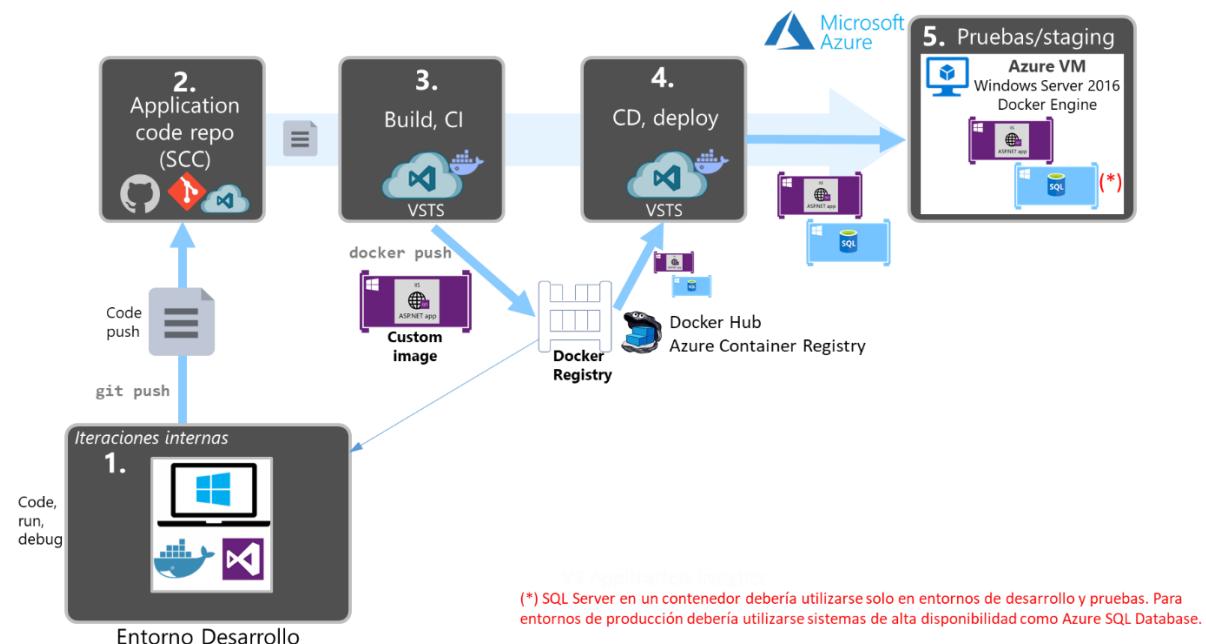


Figura 5-6. Despliegue a una Azure VM desde procesos CI/CD en Visual Studio Team Services

Azure VMs para Contenedores Windows

Las máquinas virtuales Azure para Contenedores Windows son simplemente máquinas virtuales basadas en Windows Server 2016, Windows 10 o versiones posteriores, ambas con la configuración de Docker Engine. En la mayoría de los casos, se usará Windows Server 2016 en las máquinas virtuales de Azure.

Actualmente, Azure proporciona una máquina virtual llamada **Windows Server 2016 con Contenedores**. Se puede usar esta máquina virtual para probar la nueva característica de Contenedor de Windows Server, ya sea con Windows Server Core o Windows Nano Server. Las imágenes del sistema operativo contenedor están instaladas, y por lo tanto la máquina virtual está lista para usar con Docker.

Beneficios

Aunque los Contenedores Windows se pueden desplegar a máquinas virtuales *on-premises* de Windows Server 2016, cuando se despliega a Azure, se comienza de una forma más sencilla, con máquinas virtuales "listas para usar" de Windows Server Container. También se obtiene una ubicación común en línea a la que pueden acceder los *testers*, y escalabilidad automática a través de los *sets* de escalado de Azure VM.

Siguientes pasos

Explorar el siguiente contenido para mayor detalle en la wiki de GitHub:

[https://github.com/dotnet-architecture/eShopModernizing/wiki/03.-How-to-deploy-your-Windows-Containers-based-app-into-Azure-VMs-\(Including-CI-CD\)](https://github.com/dotnet-architecture/eShopModernizing/wiki/03.-How-to-deploy-your-Windows-Containers-based-app-into-Azure-VMs-(Including-CI-CD))

Tutorial 4: Desplegar aplicaciones basadas en Contenedores Windows a Kubernetes en Azure Container Service

Disponibilidad del tutorial técnico paso-a-paso

El tutorial técnico completo está disponible en la wiki del repo GitHub de eShopModernizing:

[https://github.com/dotnet-architecture/eShopModernizing/wiki/04.-How-to-deploy-your-Windows-Containers-based-apps-into-Kubernetes-in-Azure-Container-Service-\(Including-C-CD\)](https://github.com/dotnet-architecture/eShopModernizing/wiki/04.-How-to-deploy-your-Windows-Containers-based-apps-into-Kubernetes-in-Azure-Container-Service-(Including-C-CD))

Visión general

Una aplicación basada en Contenedores Windows necesitará rápidamente usar plataformas, alejándose aún más de las VM de IaaS. Esto es necesario para lograr fácilmente una alta y mejor escalabilidad automatizada, y para una mejora significativa en despliegues automatizados y el versionado. Se pueden lograr estos objetivos utilizando el orquestador [Kubernetes](#), disponible en [Azure Container Services](#).

Objetivos

El objetivo de este tutorial es aprender a desplegar una aplicación basada en Contenedores Windows para Kubernetes (también llamada K8) en Azure Container Service. Desplegar Kubernetes desde cero es un proceso de dos pasos:

1. Desplegar un clúster de Kubernetes a Azure Container Service.
2. Desplegar la aplicación y los recursos relacionados en el clúster de Kubernetes.

Escenarios

Escenario A: Desplegar directamente a un clúster de Kubernetes desde un entorno de desarrollo

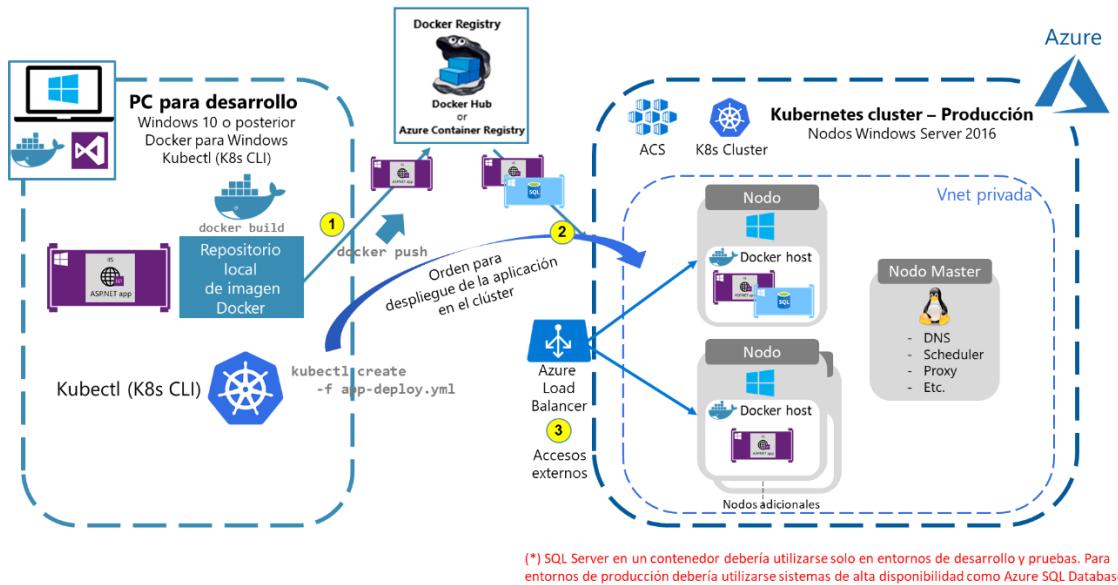


Figura 5-7. Despliegue directo a un cluster Kubernetes desde un entorno de desarrollo

Escenario B: Desplegar a un cluster Kubernetes desde procesos CI/CD em Team Services

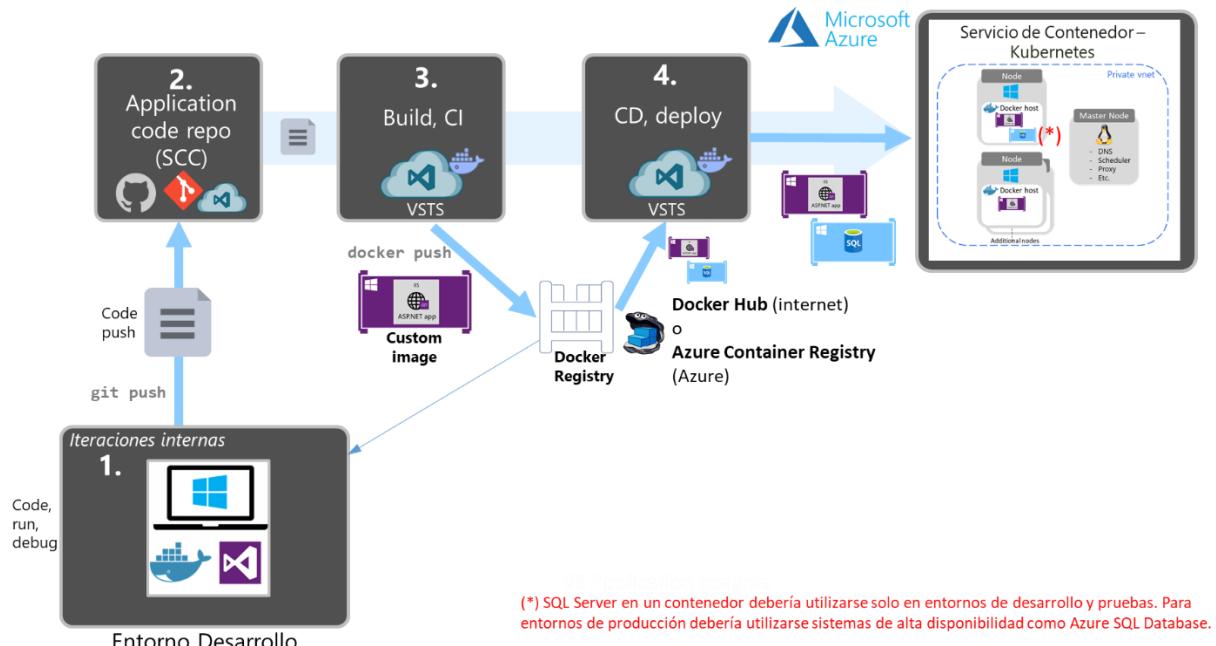


Figura 5-8. Despliegue a un cluster Kubernetes desde procesos CI/CD en Team Services

Beneficios

El despliegue a un clúster en Kubernetes tiene muchos beneficios. El mayor beneficio es que se obtiene un entorno preparado para producción en el que se puede ampliar la aplicación en función de la cantidad de instancias de contenedor que se deseen utilizar (escalabilidad interna en los nodos existentes) y en función de la cantidad de nodos o VMs en el clúster (escalabilidad global del clúster).

Azure Container Service optimiza las herramientas y tecnologías populares de código abierto específicamente para Azure. Se obtiene una solución abierta que ofrece portabilidad, tanto para los contenedores como para la configuración de la aplicación. Solo es necesario seleccionar el tamaño, la cantidad de *hosts* y las herramientas del orquestador—el servicio de contenedor maneja todo lo demás.

Con Kubernetes, los desarrolladores pueden pasar de pensar en máquinas físicas y virtuales, a planificar una infraestructura centrada en contenedores que facilite las siguientes capacidades, entre otras:

- Aplicaciones basadas en contenedores múltiples
- Replicación de instancias de contenedor y autoescalado horizontal
- Nombrar y descubrir (por ejemplo, DNS interno)
- Balanceo de carga
- Actualizaciones continuas
- Distribución de claves, contraseñas y certificados
- Comprobaciones de la salud de la aplicación

Siguientes pasos

Explorar el siguiente contenido para mayor detalle en la wiki de GitHub:

[https://github.com/dotnet-architecture/eShopModernizing/wiki/04.-How-to-deploy-your-Windows-Containers-based-apps-into-Kubernetes-in-Azure-Container-Service-\(Including-C-CD\)](https://github.com/dotnet-architecture/eShopModernizing/wiki/04.-How-to-deploy-your-Windows-Containers-based-apps-into-Kubernetes-in-Azure-Container-Service-(Including-C-CD))

Tutorial 5: Desplegar aplicaciones basadas en Contenedores Windows a Azure Service Fabric

Disponibilidad del tutorial técnico paso-a-paso

El tutorial técnico completo está disponible en la wiki del repo GitHub de eShopModernizing:

[https://github.com/dotnet-architecture/eShopModernizing/wiki/05.-How-to-deploy-your-Windows-Containers-based-apps-into-Azure-Service-Fabric-\(Including-CI-CD\)](https://github.com/dotnet-architecture/eShopModernizing/wiki/05.-How-to-deploy-your-Windows-Containers-based-apps-into-Azure-Service-Fabric-(Including-CI-CD))

Visión general

Una aplicación basada en Contenedores Windows necesitará rápidamente usar plataformas, alejándose aún más de las VM de IaaS. Esto es necesario para lograr fácilmente una alta y mejor escalabilidad automatizada, y para una mejora significativa en despliegues automatizados y versionado. Se pueden lograr estos objetivos utilizando el orquestador Azure Service Fabric, que está disponible en la nube de Azure, pero también está disponible para usarlo *on-premises*, o incluso en una nube pública diferente.

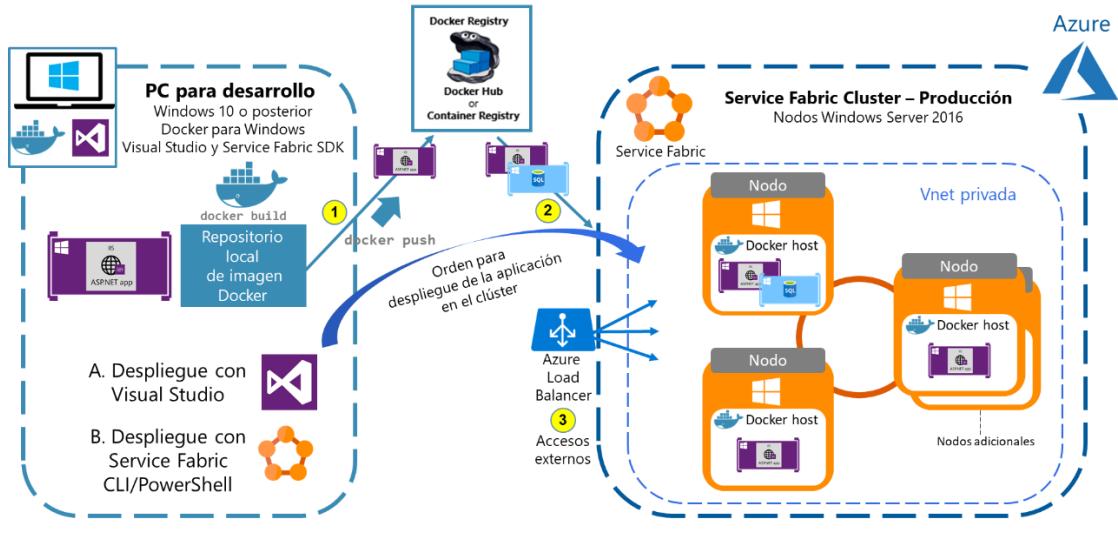
Objetivos

El objetivo de este tutorial es aprender a desplegar una aplicación basada en Contenedores Windows para Kubernetes (también llamada K8) a un cluster de Service Fabric en Azure. Desplegar a Service Fabric desde cero es un proceso de dos pasos:

1. Desplegar un clúster de Service Fabric a Azure (u otro entorno diferente).
2. Desplegar la aplicación y los recursos relacionados al clúster de Service Fabric.

Escenarios

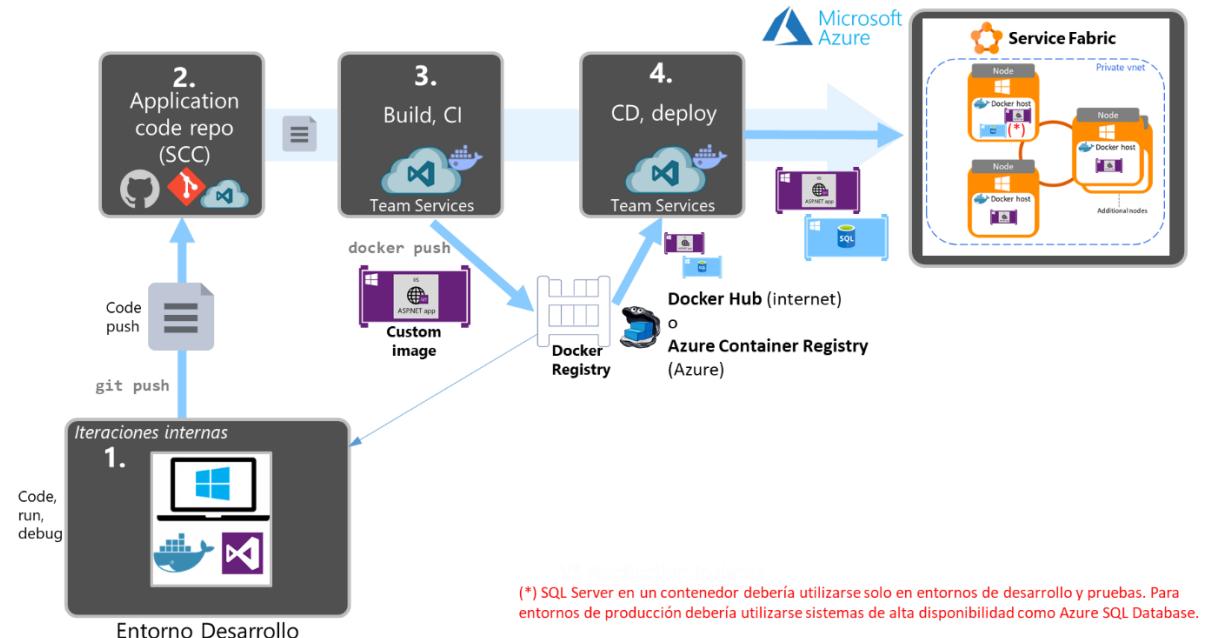
Escenario A: Desplegar directamente a un clúster de Service Fabric desde un entorno de desarrollo



(*) SQL Server en un contenedor debería utilizarse solo en entornos de desarrollo y pruebas. Para entornos de producción debería utilizarse sistemas de alta disponibilidad como Azure SQL Database.

Figura 5-9. Despliegue directo a un cluster Service Fabric desde un entorno de desarrollo

Escenario B: Desplegar a un clúster de Service Fabric desde procesos de CI/CD en Team Services



(*) SQL Server en un contenedor debería utilizarse solo en entornos de desarrollo y pruebas. Para entornos de producción debería utilizarse sistemas de alta disponibilidad como Azure SQL Database.

Figura 5-10. Despliegue a un cluster Service Fabric desde procesos CI/CD en Visual Studio Team Services

Beneficios

Los beneficios de desplegar en un clúster de Service Fabric son similares a los beneficios del uso de Kubernetes. Una diferencia, sin embargo, es que Service Fabric es un entorno de producción en estado muy maduro para aplicaciones de Windows en comparación con Kubernetes, que estaba en *preview* para Contenedores Windows hasta principios del otoño de 2017. (Kubernetes es un entorno más maduro para Linux).

La principal ventaja de utilizar Azure Service Fabric es que se obtiene un entorno preparado para producción en el que se puede escalar la aplicación en función de la cantidad de instancias de contenedor que se deseen usar (escalabilidad interna en los nodos existentes), y en función de la cantidad de nodos o máquinas virtuales en el clúster (escalabilidad global del clúster).

Azure Service Fabric ofrece portabilidad tanto para los contenedores como para la configuración de la aplicación. Se puede tener un clúster de Service Fabric en Azure o instalarlo *on-premises* en el propio centro de datos. Incluso se puede instalar un clúster de Service Fabric en una nube diferente, como [Amazon AWS](#).

Con Service Fabric, los desarrolladores pueden pasar de pensar en máquinas físicas y virtuales a planificar una infraestructura centrada en contenedores que facilite las siguientes capacidades, entre otras:

- Aplicaciones basadas en contenedores múltiples
- Replicación de instancias de contenedor y autoescalamiento horizontal
- Nombrar y descubrir (por ejemplo, DNS interno)
- Balanceo de carga
- Actualizaciones continuas
- Distribución de claves, contraseñas y certificados
- Comprobaciones de la salud de la aplicación

Las siguientes capacidades son exclusivas de Service Fabric (en comparación con otros orquestadores):

- Capacidad de servicios *stateful* (con estado en memoria y persistido en el cluster), a través del modelo de aplicación de Reliable Services.
- Patrón de actores, a través del modelo de aplicación de Reliable Actors.
- Desplegar procesos *bare-bone*, además de Contenedores Windows o Linux.
- Actualizaciones progresivas y controles de estado.

Siguientes pasos

Explorar el siguiente contenido para mayor detalle en la wiki de GitHub:

[https://github.com/dotnet-architecture/eShopModernizing/wiki/05.-How-to-deploy-your-Windows-Containers-based-apps-into-Azure-Service-Fabric-\(Including-CI-CD\)](https://github.com/dotnet-architecture/eShopModernizing/wiki/05.-How-to-deploy-your-Windows-Containers-based-apps-into-Azure-Service-Fabric-(Including-CI-CD))

Conclusiones

Puntos clave

- Las soluciones basadas en contenedores brindan finalmente beneficios de ahorro de costes. Los contenedores son una solución para los problemas de despliegue porque eliminan la fricción causada por la falta de dependencias en los entornos de producción. Al eliminar estos problemas, mejora las operaciones de desarrollo/pruebas, DevOps y producción de manera significativa.
- Docker se está convirtiendo en el estándar de facto en la industria de los contenedores. Docker cuenta con el respaldo de los proveedores más destacados en los ecosistemas de Linux y Windows, incluido Microsoft. En el futuro, proporcionará el entorno más amplio y completo para modernizar las aplicaciones .NET Framework existentes con los Contenedores Windows y los servicios de infraestructura de Azure. El contenedor Docker se está convirtiendo en la unidad estándar de despliegue para cualquier aplicación o servicio basado en servidor.
- Para los entornos de producción, se debe usar un orquestador (como Service Fabric o Kubernetes) para alojar aplicaciones escalables basadas en Contenedores Windows.
- Las Azure VMs que alojan contenedores son una forma rápida y sencilla de crear pequeños entornos de desarrollo/pruebas en la nube.
- Azure SQL Database Managed Instance es la opción recomendada "por defecto" al migrar las bases de datos relacionales, desde aplicaciones existentes a Azure.
- Visual Studio 2017 e Image2Docker son herramientas fundamentales para que se puedan comenzar a modernizar las aplicaciones .NET existentes con Contenedores Windows, acelerando la curva de aprendizaje inicial.
- Cuando se establezcan aplicaciones contenerizadas en producción, siempre se tendrán que sustentar en una cultura DevOps y en herramientas DevOps para procesos de CI/CD, como Visual Studio Team Services o Jenkins.
- Microsoft Azure proporciona el entorno más amplio y completo para modernizar aplicaciones .NET Framework existentes, con Contenedores Windows, infraestructura en la nube y servicios PaaS.