

게임 서버 개발자를 위한 Windows 컨테이너 입문

DEVSISTERS

남정현



The slide features decorative geometric shapes. On the left, there are several overlapping yellow squares and diamonds of various sizes. On the bottom right, there are overlapping blue squares and diamonds. The main content is centered on a white background.

시작하기 전에

- 이 세션은 Docker에 대한 기본적인 사용법을 이해하고 계신 분들을 대상으로 합니다.
- Docker 튜토리얼 추천:

pyrasis.com/Docker/Docker-HOWTO




스피커 소개

남정현 (jeonghyun.nam@devsisters.com)

- **DEVSISTERS** DevOps Engineer
 - 닷넷데브 운영진
 - Microsoft MVP (2009 - 2020)
- 




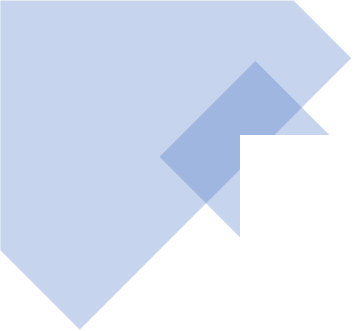
Agenda

- Real World Story
 - Windows Container Intro
 - Developing Win32 Container
 - Windows Kubernetes
- 



Real World Story

- 파티파티: 데코플레이는 Windows 컨테이너와 클라우드 상에서 실행된 게임입니다.
 - Windows Server Core, Public Cloud 환경 위에서 인프라 구축 및 실행했습니다.
 - 2020년 2월 13일부터 약 일주일간 CBT로 운영하였습니다.
 - YouTube 겜조티비의 리뷰 영상으로 게임에 대해 간단히 소개합니다.
- 



DEVSISTERS와 Windows Workload

- DEVSISTERS에서는 게임 서버 개발과 테스트 자동화를 위하여 Kubernetes를 성공적으로 도입했습니다.
 - <https://www.slideshare.net/seungyongoh3/ndc17-kubernetes>
- 2019년 8월부터는 관리형 Kubernetes 클러스터와 Windows 컨테이너 지원을 도입하여 사용 중입니다.

DEVSISTERS와 Windows Workload (Cont.)

- Windows 컨테이너 지원에 대한 R&D는 2017년 하반기부터 시작된 여정을 2019년 2월에 완료하였습니다.
 - <https://github.com/kubernetes/kubernetes/issues/65163>
 - <https://github.com/kubernetes/kubernetes/issues/66947>
- Microsoft, Windows Kubernetes SIG와 협력하여 활발하게 토론을 진행하고 있습니다.

구축 목표, 과정, 성과

구축 목표

- Windows와 Linux Pod
동시 사용
- 동일 서브넷에서 서로
다른 노드 OS 구동

구축 과정 및 성과

- 2018년 여름에
한시적으로 하이브리드
클러스터 운영
- 그 과정에서 발견된
2개의 이슈를 Windows
SIG 팀과 협업하여 해결
- 구축 및 운영 노하우
획득

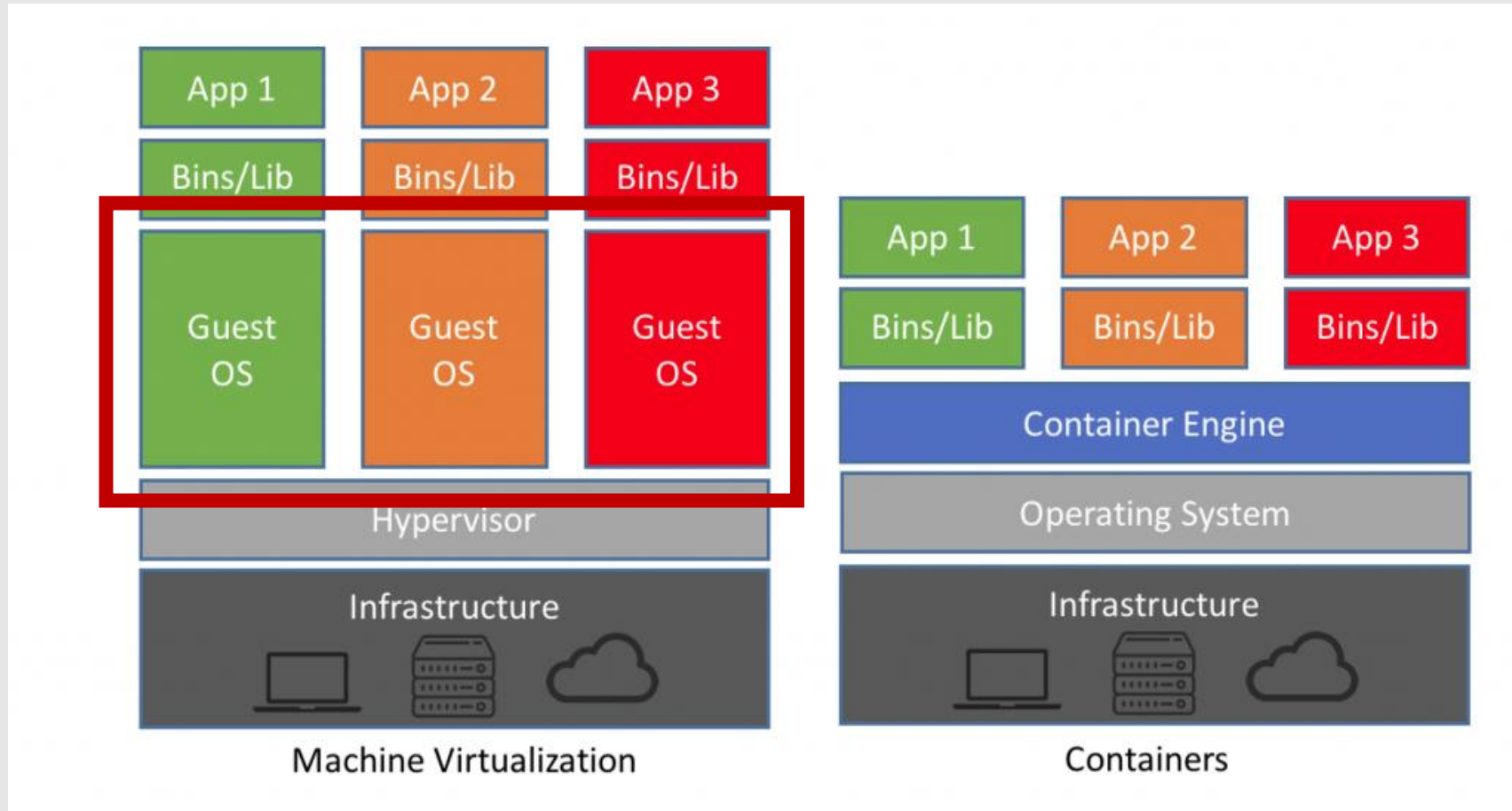
구축 결과

- KOPS 클러스터에
Windows 노드 추가
- Windows Server 2019
대상
- MS SDN 스크립트로부터
설치 자동화를 위한
자체 Tool 개발



Windows Container Intro

Container vs. Virtual Machine



Container vs. Virtual Machine (Cont.)

- 일반적으로 VM의 이점을 Container도 누릴 수 있습니다.
 - 반면 이것이 많은 혼란과 안티 패턴을 일으킬 수도 있습니다.
- 세세한 기능들에 대해서는 잠시 잊으세요.
 - VM은 컴퓨터의 상태 그 자체를 표현하는 것에 초점을 맞추고 있어 그 자체로 완결성이 있습니다.
 - Container는 그 안에 담길 애플리케이션에만 초점을 맞춥니다. 나머지는 컨테이너 엔진과 오케스트레이션 도구에 의해 정의됩니다.

컨테이너화를 하는 이유

일관성

빠른 개발

좀 더 많은
반복 주기

손쉬운 재현

클라우드
플랫폼 중립성

환경 불변성

효율성

서비스 간의
분리

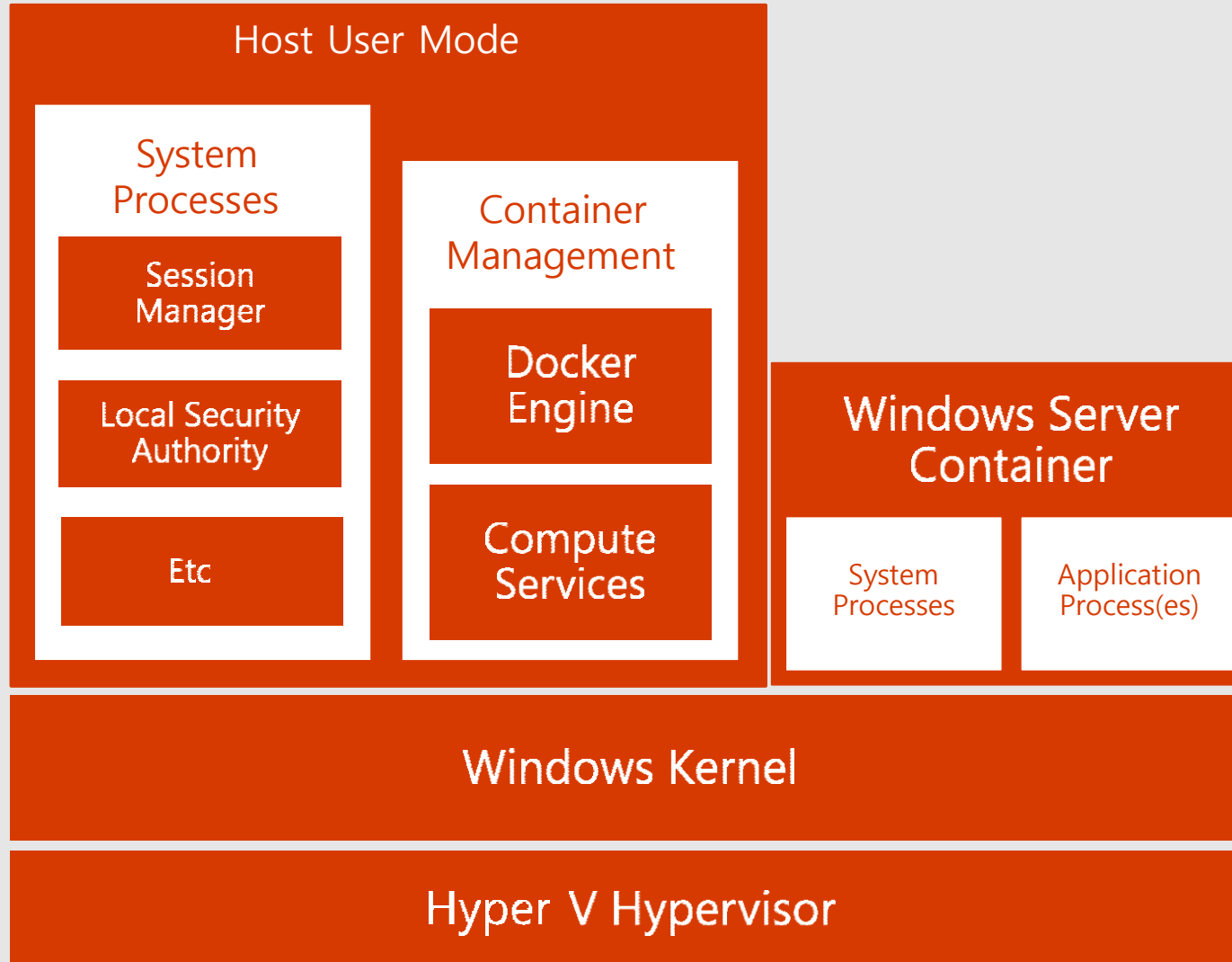
높은 가용성

확장성

대규모
테스트 지원

저렴한 비용

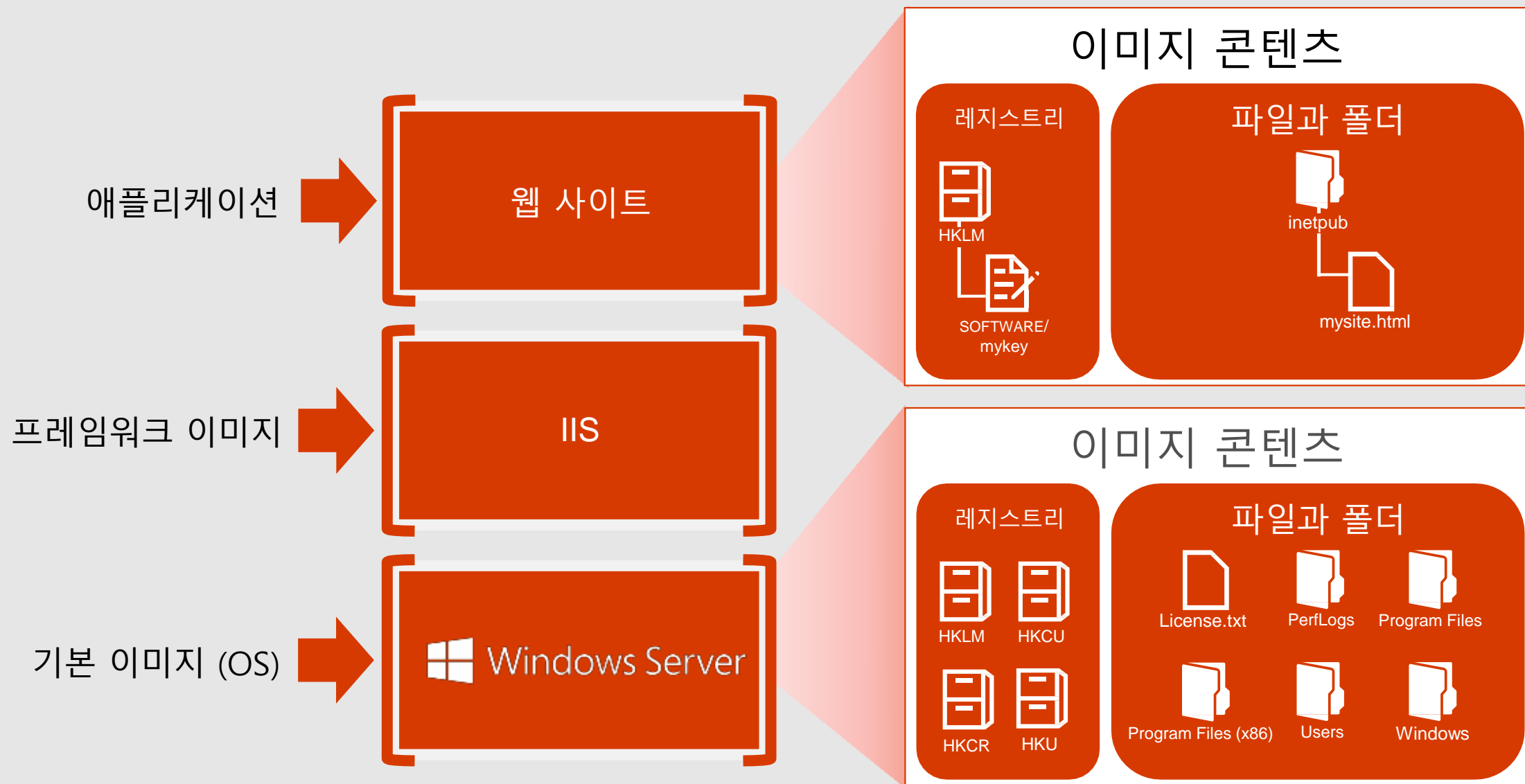
Windows 서버 컨테이너



Windows 서버 컨테이너 (Cont.)

- OS 수준의 가상화이므로 하이퍼바이저 불필요
 - 커널이 컨테이너 내부의 모든 프로세스를 관리
 - 기본 이미지로부터 차등분만 이미지로 저장
- VM을 사용하는 것 보다 훨씬 더 조밀한 서비스 제공 가능
- 단, 동일 커널을 사용하여 발생할 수 있는 이슈가 존재
 - 이것은 리눅스 기반 컨테이너에서도 동일한 이슈
- Shared Kernel 방식의 컨테이너 실행 가능 개수는 무제한

컨테이너 이미지



자동화된 이미지 빌드

Docker Build 명령과 Dockerfile

자동화된 이미지 빌드 방법
"docker build" 명령으로 실행
단계별 명령 수행 결과가 캐시됨
Docker Hub에 연동됨

Dockerfile 예제

IIS

```
FROM mcr.microsoft.com/windows/servercore  
RUN powershell -Command Add-WindowsFeature Web-Server
```

웹 사이트

```
FROM mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019  
ADD mysite.htm inetpub\mysite.htm
```

My Website

IIS

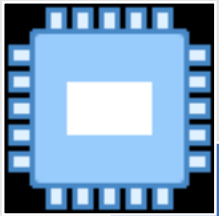
 Windows Server

Windows 컨테이너의 버전 선택

| Host > Container V | Win Server 2016 | Win 10 Creators Update 1703 | Win Server 1803 | Win 10 Fall Creators Update 1803 | Win Server 2019 | Win 10 2018 April Update 1809 |
|-----------------------|-----------------------|--------------------------------------|-----------------------|--|-----------------------|---|
| Win Server 2016 | Process Hyper-V | Hyper-V | Hyper-V | Hyper-V | Hyper-V | Hyper-V |
| Win Server 1803 | 사용 불가 | 사용 불가 | Process Hyper-V | Hyper-V | Hyper-V | Hyper-V |
| Win Server 2019 | 사용 불가 | 사용 불가 | 사용 불가 | 사용 불가 | Process Hyper-V | Hyper-V |

- 커널 구조의 차이로 도입
 - Process 방식, Hyper-V 방식
- Windows 10에서 Windows Container를 실행 시
 - OS 버전과 컨테이너 이미지 버전 일치 필요
- 버전 간 호환성에 대하여
 - <https://bit.ly/2JSTo5A>

Windows 컨테이너 이미지의 선택



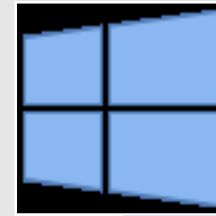
Nano Server

- 빠른 프로비저닝
- Windows OS 종속성이 없는 애플리케이션에 최적화 (예: Go, .NET Core 등)
- 작은 이미지 크기 (80~100MB)



Server Core

- 호환성을 중시
- 완전한 .NET Framework 지원
- 대부분의 Windows 백엔드 애플리케이션에 적합
- 약 1.4GB 이미지 크기



Windows

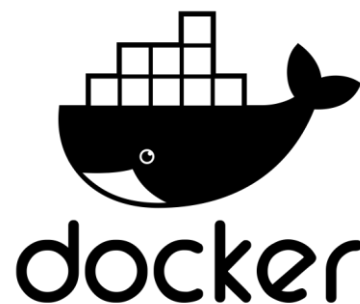
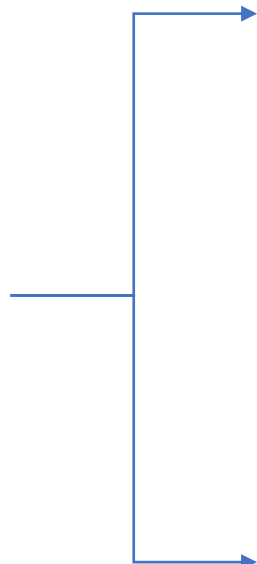
- GUI 애플리케이션 자동화에 적합
- Windows OS 구성 요소의 대부분을 지원 (DirectX 포함)
- 약 3.5GB 이미지 크기

Docker 프로젝트의 관계도



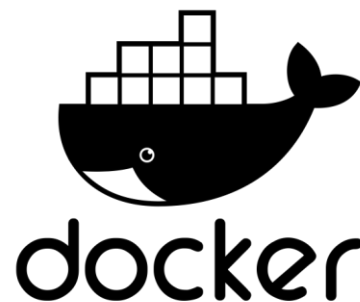
Moby Project

컨테이너 플랫폼을 만드는
핵심 구성 요소를
조립하기 위한 오픈 소스
프레임워크



Docker **Enterprise Edition**

구독 기반의 상용 컨테이너
플랫폼 및 서비스 제공



Docker **Community Edition**

무료로 제공되는 커뮤니티
중심의 제품

Windows에서의 컨테이너 환경



Docker Toolbox for Windows

Windows 10 1607보다 낮은 버전의 Windows에서 리눅스 컨테이너를 사용하기 위한 목적으로 설치 가능한 개발자용 도구



Docker Desktop for Windows (Community Edition)

네이티브 Windows 컨테이너 개발 및 리눅스 컨테이너 개발을 모두 지원하는 개발자용 도구

Windows 10 1주년 업데이트 이상 또는 **데스크톱 경험을 포함한 Windows Server 1607 이상**에서 사용할 수 있는 개발자용 도구

Windows에서의 컨테이너 환경 (Cont.)

- Docker Enterprise Edition for Windows Server
 - **Windows Server 2016/1607 이상의 OS에 대해 제공되는 프로덕션용 Docker**
 - 리눅스 컨테이너에 대한 실험적 지원이 제공되지만, Windows 컨테이너 전용으로 보는 것이 맞음
 - Microsoft 측에서 패키지를 제공함

Windows 10 개발 환경 구축



최신 버전의 Docker Desktop for Windows 설치



OS 버전 확인 후 일치하는 Base Image 선택

일치하지 않는 버전을 선택하는 경우 Hyper-V 가상화 필수

일치하는 경우 프로세스 방식으로 컨테이너 실행

Daemon

Configure the Docker daemon by typing a [configuration file](#).

☒ Advanced

This can prevent Docker from starting. Use

```
{
  "registry-mirrors": [],
  "insecure-registries": [],
  "debug": true,
  "experimental": true,
  "exec-opts": [
    "isolation=process"
  ]
}
```

Docker will restart when applying these settings.

Docker Daemon 설정

- 옆의 그림과 같이 “exec-opts” 설정을 추가하여 Isolation Mode를 Process로 변경 (중요)
- <https://github.com/moby/moby/pull/38000>

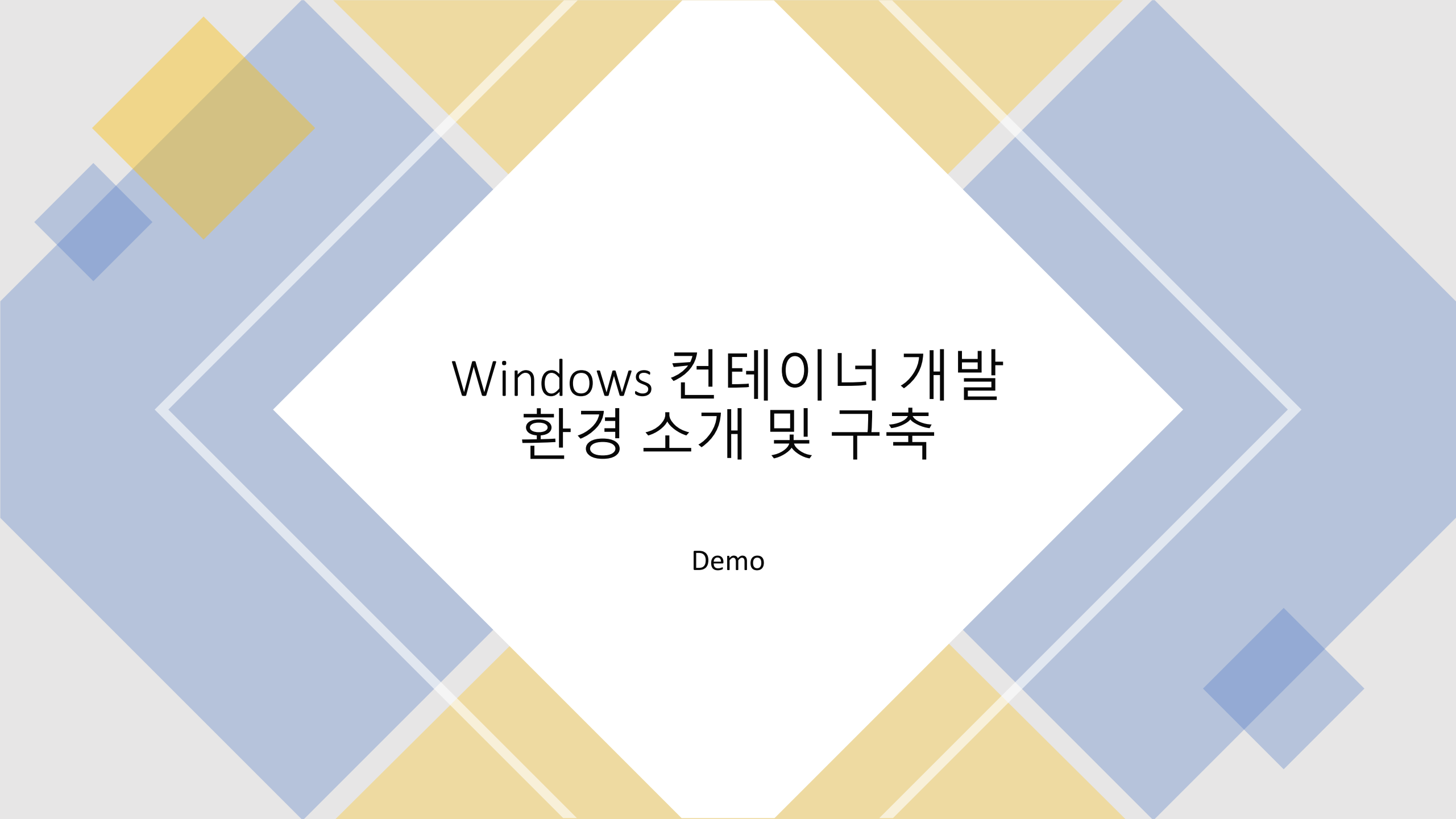
Windows 10 개발 환경 구축 (Cont.)

가능한 Hyper-V 격리는 개발 환경에서는 피하는 것을 권장

- 프로세스 격리 방식보다 안전함
- 개발/테스트 과정에서 매우 큰 오버헤드를 수반하고 매우 느림
- 특히 Dockerfile 빌드 시 극단적인 성능 저하가 따름

서버의 버전과 일치하는 Windows 10 OS 사용 권장

- 불가피한 경우 별도 VM 구축 등을 고려해볼 수 있음



Windows 컨테이너 개발 환경 소개 및 구축

Demo

컨테이너 빌드 전략

- 최초 컨테이너 이식: Core에서 작업
- 메모리 크기, 실행 성능 최적화가 가능하고, OS 의존성이 낮다면
 - 다중 스테이지 빌드를 활용하여 필요한 작업은 Core에서 모두 처리
 - 최종적으로 만들어지는 Artifact를 Nano로 복사하고 시스템 구성
- 실제 사례: <https://github.com/rkttu/python-nanoserver>



몇 가지 허들

- 애플리케이션 호환성
 - 32비트 바이너리를 사용하거나 GUI 의존성이 있는지 확인합니다.
- 라이프사이클
 - NT 서비스를 Docker의 라이프사이클 안으로 포함시켜 Docker가 인지할 수 있어야 합니다.
- 로그 데이터 수집
 - Docker는 콘솔 출력을 이용하여 모든 메트릭, 디버깅, 로그 정보를 바깥으로 내보냅니다.
- .NET에 관한 것
 - 애플리케이션 시작 시간 단축을 위한 최적화 과정이 필요합니다.
- 데이터베이스에 관한 것



허들 극복하기 - 애플리케이션 호환성

- 만약 32비트 애플리케이션을 컨테이너화해야 한다면?
 - Windows Server Core 컨테이너만 사용 가능
 - Nano Server에는 Windows-on-Windows 기능이 없어서 32비트 앱은 실행 불가함
- 만약 GUI 기능을 사용해야 한다면?
 - Windows Container는 GUI, RDP 등 사용자 상호 작용에 관련된 기능이 전혀 없음
 - 완전한 UI 자동화 테스트가 필요한 경우 Full Container를 사용할 수는 있음

허들 극복하기 - NT 서비스

- <https://github.com/microsoft/IIS.ServiceMonitor>
 - IIS 서비스의 시작과 끝을 컨테이너의 시작과 끝에 맞추는 도구
 - IIS만이 아니라 일반적인 NT 서비스에 대해서도 쓸 수 있음
 - 컨테이너의 환경 변수들을 IIS 프로세스와 동기화함



허들 극복하기 - .NET Framework

- <https://github.com/Microsoft/dotnet-framework-docker/pull/27>
 - 애플리케이션의 초기 시작 시간 단축을 위하여 도입된 패치
 - 하지만 IIS 컨테이너에서는 이 부분이 누락되어 있음



허들 극복하기 - 로깅

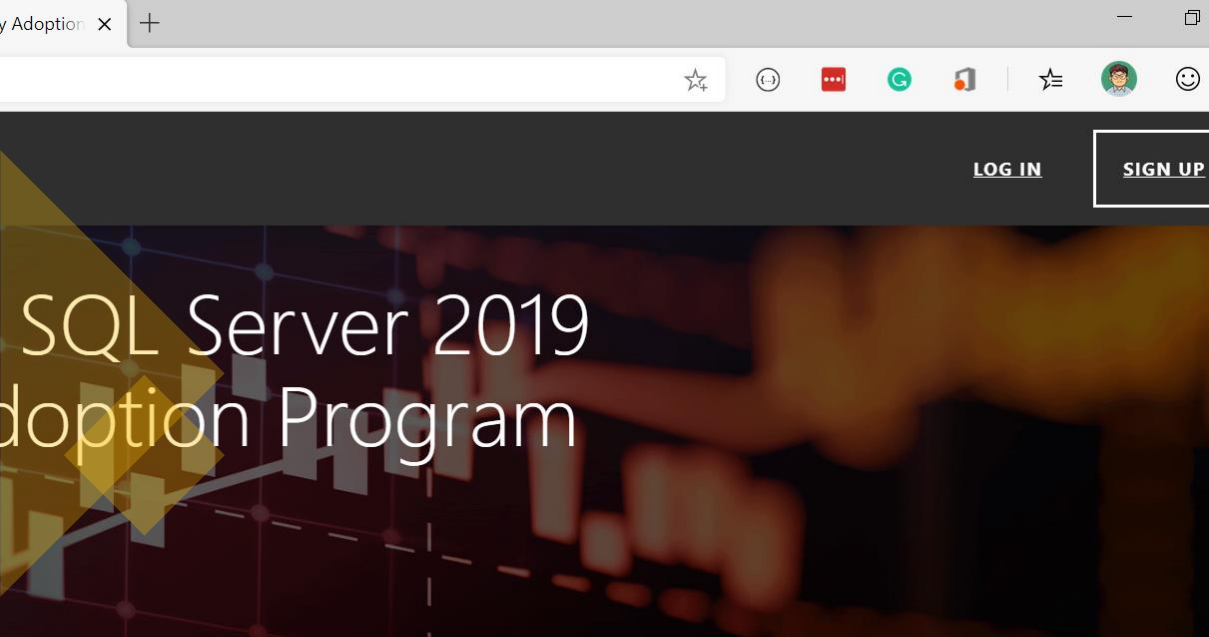
- <https://github.com/microsoft/windows-container-tools/tree/master/LogMonitor>
 - 콘솔을 제외한 파일 로그, 이벤트 로그, ETW로 산재되는 로그 기록
 - 이 모든 것을 한 번에 모아서 콘솔로 내보내 줄 도구가 최근에 공개됨





데이터베이스에 관하여

- Production에서 사용할 수 있는 컨테이너 기반 MS SQL Server는 리눅스 버전이 유일
- 기술적으로 Express Edition을 개발/테스트 목적으로 사용할 수는 있으나 Production 품질 보장이 되지 않음
- MS SQL Server 2019를 Production Windows Container로 체험해볼 수 있는 프리뷰가 진행 중
- <https://cloudblogs.microsoft.com/sqlserver/2019/07/01/sql-server-2019-on-windows-containers-now-in-early-adopters-program/>



When you have your data, you want performance and security that won't get in the way of your business. With traditional database management systems, Microsoft continuously improves its products to help you manage your expanding data world.


Integrate your data with Spark and HDFS into a single combined cluster for big scale and analytics. Connect to your data from a variety of sources, including Oracle, Teradata, other SQL Servers, and other ODBC compliance data sources.

데이터베이스에 관하여 (Cont.)

- 현재 MSSQL의 경우 별도 호스트에서 실행하거나, Kubernetes에 리눅스 컨테이너로 클러스터를 만드는 것이 권장됨
- 추후 Windows Container용 제품이 출시될 것으로 기대됨

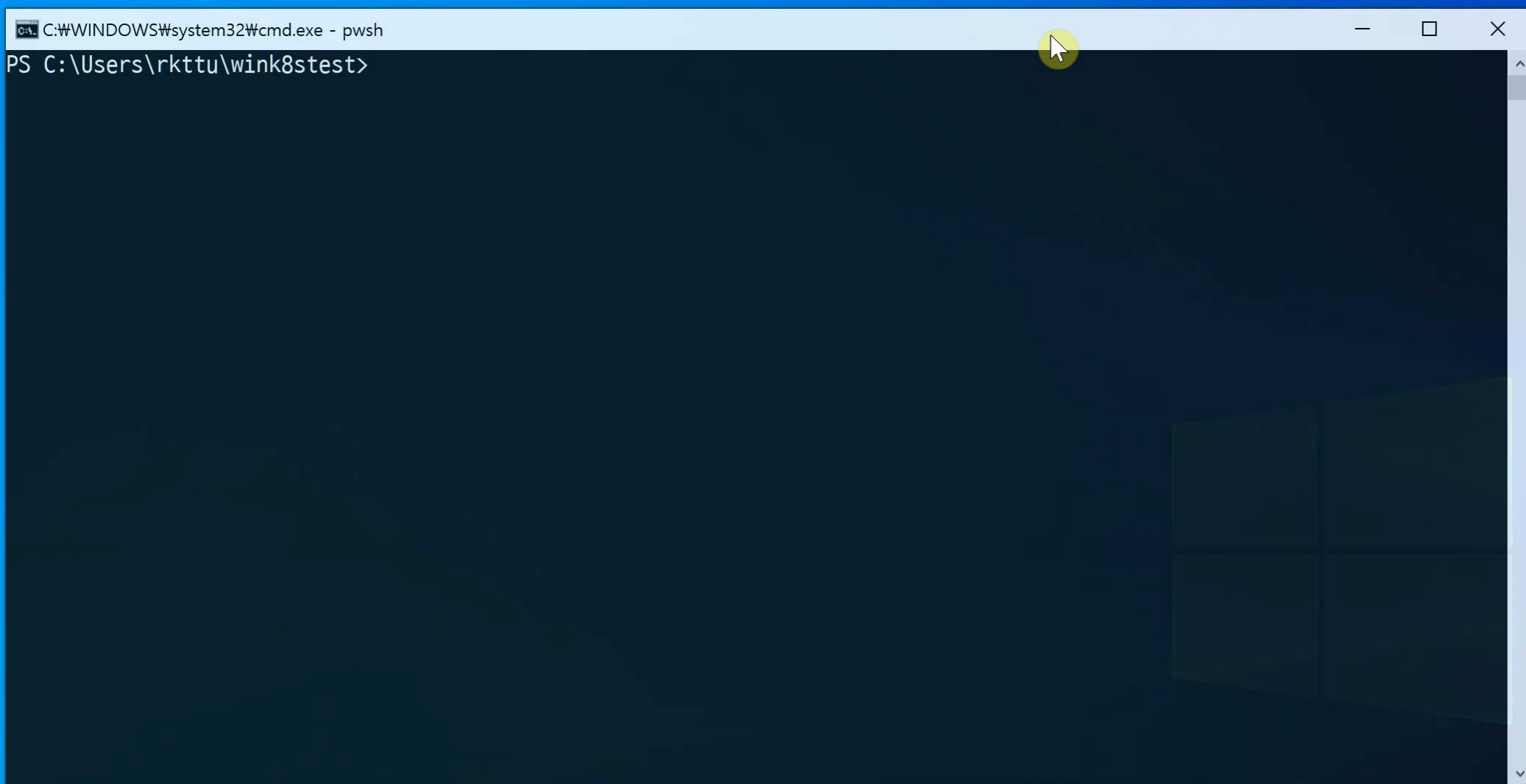


What's Next?

- 애플리케이션을 컨테이너화하면 Kubernetes로 배포 가능
 - 최적의 클러스터 노드를 찾아 리소스를 자동 배정하는 것은 물론
 - 애플리케이션 자체 문제는 물론 클러스터에 문제가 생겨도 자동으로 새 인스턴스를 띄워 주고 자동으로 연결시켜줍니다.
 - 또한 리눅스와 윈도우 애플리케이션을 원래 하 나였던 것처럼 섞어서 사용할 수 있게 됩니다.
- 

Deploy Hybrid Workload

Demo



The slide features decorative geometric shapes in the corners. The top-left corner has several overlapping yellow squares of various sizes. The bottom-right corner has overlapping blue squares. The main content area is white.

샘플 YAML 받아보기


- <https://gist.github.com/rkttu/e30542d75b8e48d0cca1eec6eb0cf86b>
- Hybrid Workload에서 중요한 것은 Node Selector를 통한 적절한 노드 선택.
- Windows 노드에 Linux 컨테이너를 배정하거나, 반대의 상황이 발생하면 서비스가 시작되지 못함.

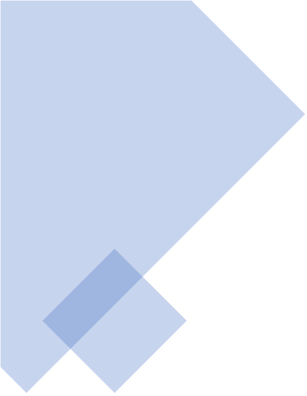


Conclusion




마무리

- 하나의 기술 스택만 사용하는 시대는 끝났습니다.
 - OS, 플랫폼, 기술에 관계없이 한 데 어우러져 하나의 서비스를 만드는 것이 대세인 시대가 되었습니다.
 - 기술도 다양성이 존중되어야 합니다.
- 



마무리 (Cont.)

- Windows 플랫폼을 선택한 고객들도 새로운 선택을 고려해야 할 때가 왔습니다.
 - Windows 개발자들도 Cloud Native를 중심으로 하는 커리어 패스를 만들 때가 왔습니다.
 - 오늘부터 시작해보세요.
- 

Docker on Windows

Second Edition

From 101 to production with Docker on Windows

Packt>
www.packt.com

Elton Stoneman

참고할 만한 자료

- <https://www.packtpub.com/virtualization-and-cloud/docker-windows-second-edition>
- 현재 한국어 번역 작업이 진행 중입니다. 올해 늦봄 출간을 목표로 하고 있습니다.

DEVSISTERS



We're Hiring!
careers.devsisters.com

Merci

謝謝

Tänan

Thank You

Danke

ありがとう

Obrigado

고맙습니다