

DEVIEW 2020

클라우드 네이티브 시대의 Windows 게임 서버

남정현 DEVSISTERS

발표자 소개

DEVIEW 2020

남정현

- DEVSISTERS DevOps 엔지니어
  - Infrastructure Team
  - 파티파티: 데코플레이
- Microsoft MVP (2009년~)
- 닷넷데브, 한국 WSL 사용자 그룹

데브시스터즈 소개

모두가 DevOps Engineer

- 서버 구조, 언어, 스택, 프로토콜 설계
- 개발부터 프로덕션까지 서버 배포
- 모니터링과 장애 대응
- 뭔가 아쉬우면 새로 만들기
- 때로는 게임 서버 분석과 디버깅까지

정말 여러 가지 일이 많아요.

데브시스터즈 소개 (Cont.)

이렇게 할 일은 많지만 사람이 많은 것은 아닙니다.

- 기술로 해결할 수 있는 것은 기술로 풀자

오픈 소스와 자동화 개발에 그래서 많은 시간을 씁니다.

CONTENTS

1.윈도우 컨테이너

1.1 왜 윈도우 컨테이너인가

DEVIEW 2020

화성인과 금성인

- 윈도우 서버 애플리케이션 개발자
- DevOps 엔지니어

새로운 OS를 배우는 부담
IOCP가 아닌 다른 기술의 사용
익숙하지 않은 도구

새로운 OS를 배우는 부담
그래픽 방식의 관리 작업
배포와 관리의 부담 증가

왜 윈도우 컨테이너인가


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

왜 윈도우 컨테이너인가 (Cont.)

컨테이너가 좋다는 건 아는데...

- 윈도우 서버 개발자가 리눅스 프로그래밍을 배울 필요 없음
- DevOps 엔지니어가 RDP 도구를 따로 쓸 필요 없음
- 컨테이너와 쿠버네티스를 통하여 하나의 시점으로 통일

윈도우 서버 개발자에게는 새로운 기회

- CI, CD 방식의 자동화된 빌드 환경 도입
- 재현 불가능한 환경이나 오류를 최소화
- 현대화된 개발 환경

왜 윈도우 컨테이너인가 (Cont.)

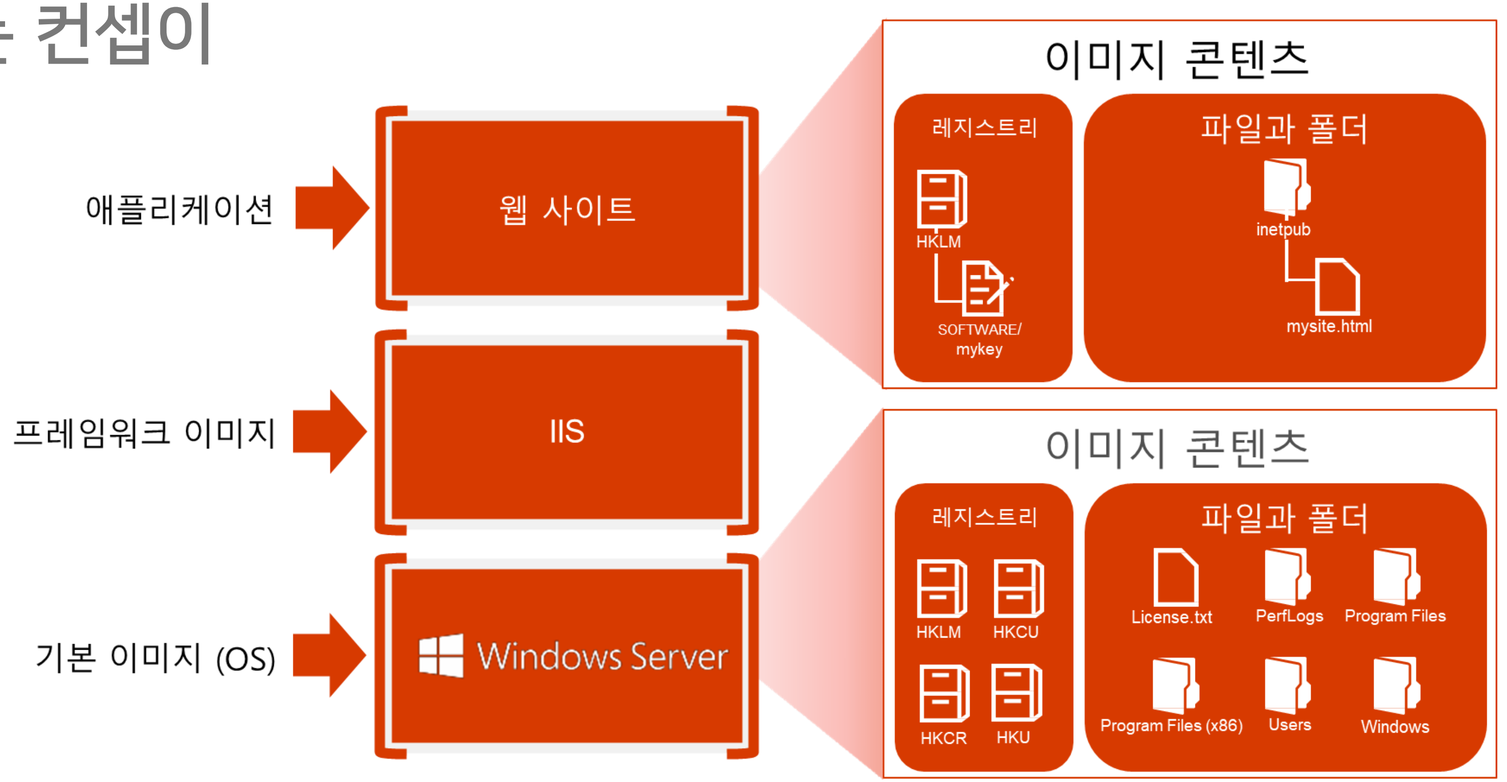
다만…

- 이미 윈도우, 리눅스 애플리케이션 개발을 잘 하고 계시다면
- 굳이 OS를 변경하는 모험을 하실 필요도, 이유도 없어요!
- OS 지원 추가는 인프라의 역할이니까요!

1.2 윈도우 컨테이너의 기본

이미지가 여러 레이어로 구성

- 기본 이미지는 Microsoft가 독점 공급
- 리눅스와 달리 레지스트리라는 컨셉이 별도 존재. 각 이미지마다 존재.

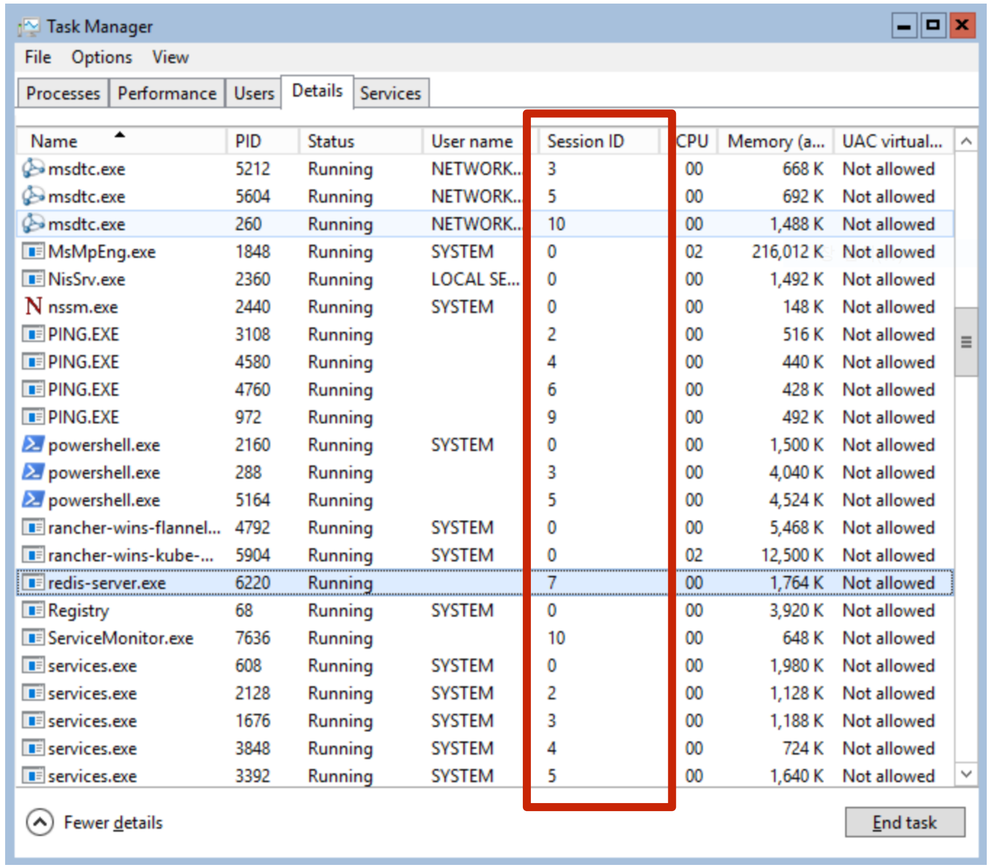


윈도우 컨테이너의 기본 (Cont.)

DEVIEW 2020

컨테이너마다 시스템 프로세스 실행

- 컨테이너마다 필수 시스템 프로세스가 구동됨
- 커널이 직접 컨테이너를 처리하는 것이 기본 방식
  - 작업 관리자에서 0 외의 세션 ID가 여럿 표시됨

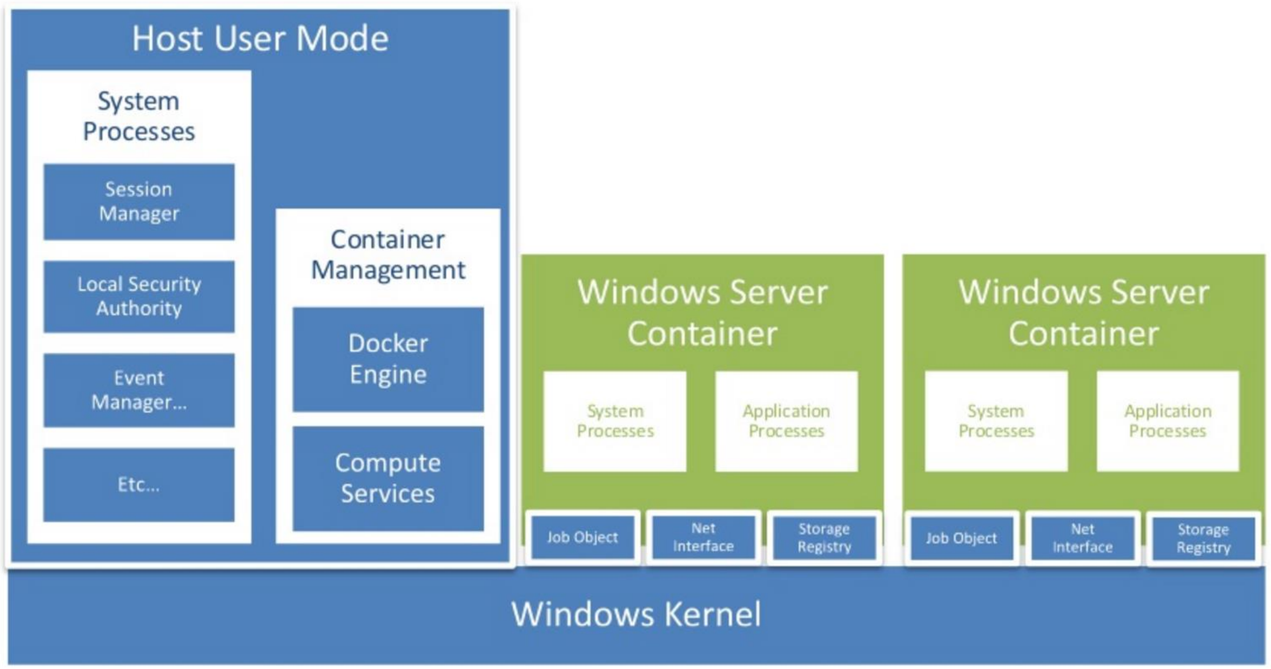


윈도우 컨테이너의 기본 (Cont.)

DEVIEW 2020

세션이 나란히 실행됨

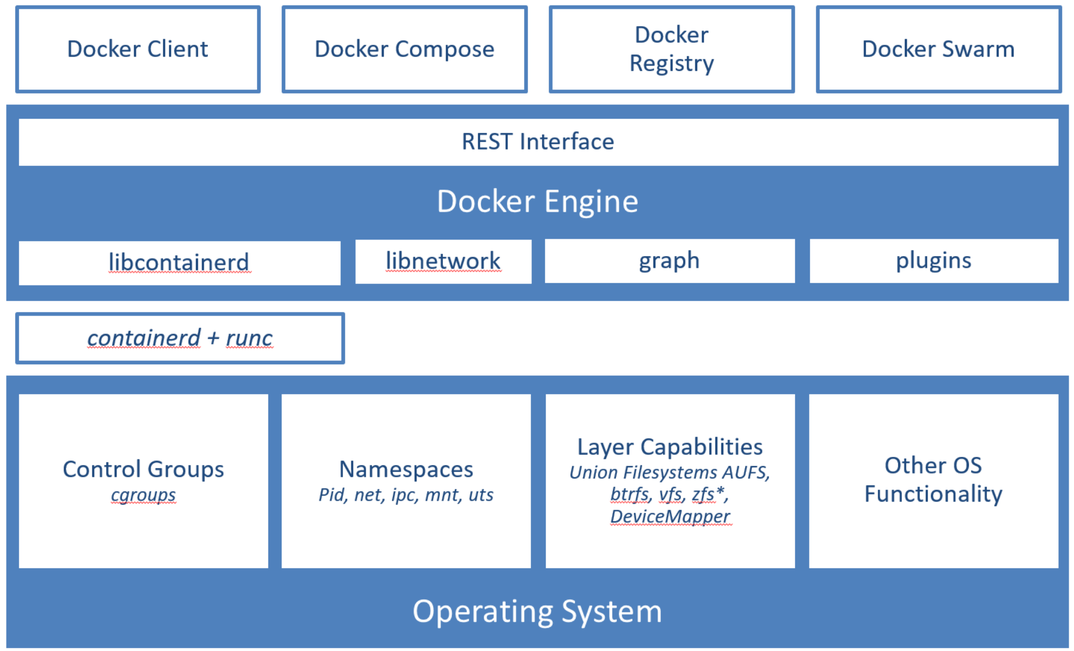
- 컨테이너 당 하나의 세션
- 세션 간에는 상호 간섭 불가
  - 서로 다른 리소스, 레지스트리
  - 파일 시스템 역시 구분됨
- 윈도우 보안 업데이트 중요성 증가



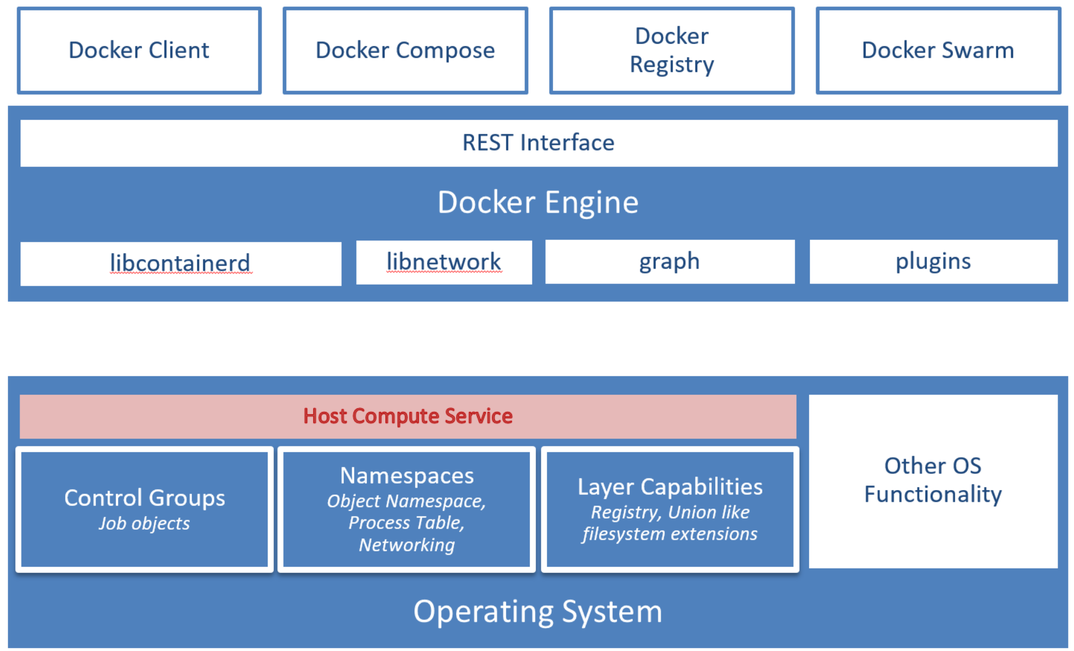
1.2 리눅스 컨테이너와 기술적 차이점

DEVIEW 2020

Architecture In Linux

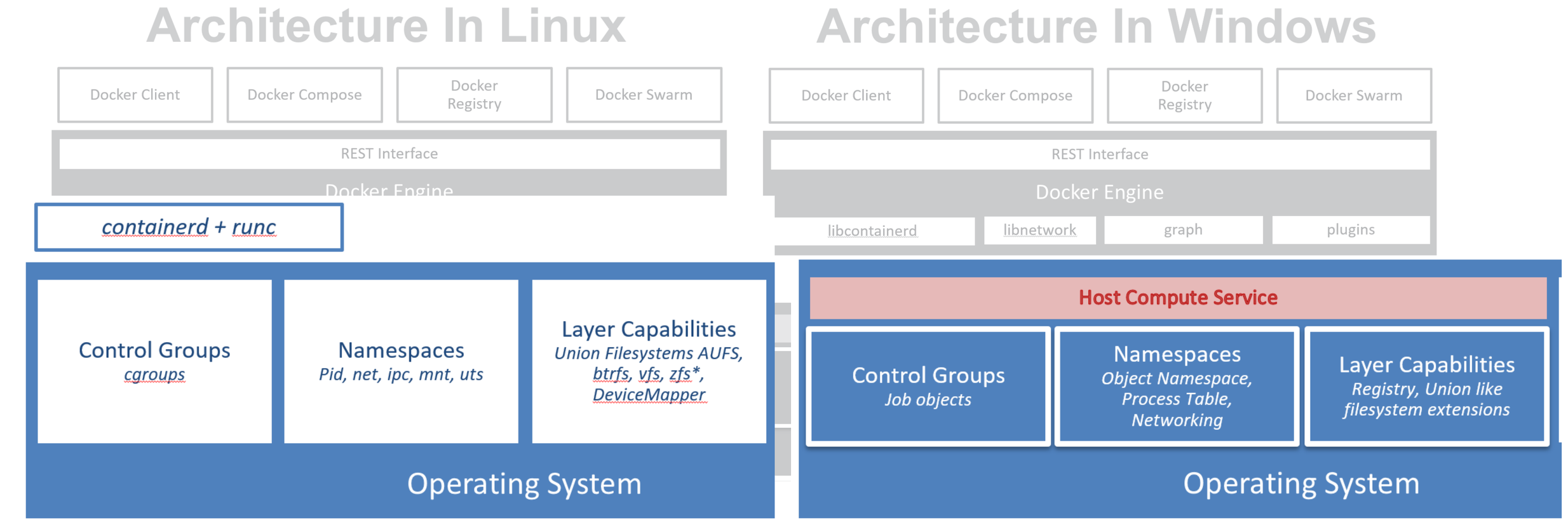


Architecture In Windows



리눅스 컨테이너와 기술적 차이점 (Cont.)

DEVIEW 2020



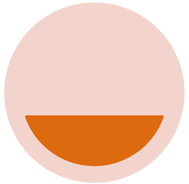
1.3 윈도 애플리케이션의 컨테이너화

베이스 이미지의 종류

- 크게 나노 서버, 서버 코어 이미지로 구분
- 나노 서버: 전통적인 윈도우 기능에 비교적 의존도가 낮은 애플리케이션에 특화
  (이미지 크기: ~1GiB)
- 서버 코어: 전통적인 윈도우 기능에 의존하는 애플리케이션의 컨테이너화에 초점
  (이미지 크기: 4~5GiB)
- 그 외 IoT, ML 컨테이너 이미지 등이 존재
  (컨테이너 호스트 OS에 따라 더 세분화 가능)

윈도우 애플리케이션의 컨테이너화 (Cont.)

DEVIEW 2020

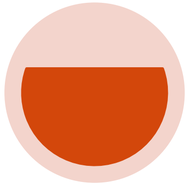


Nano Server

빠른 프로비저닝

윈도우 OS 종속성이 없는 애플리케이션에 최적화 (예: Go, .NET Core, Python 등)

~1GiB 크기



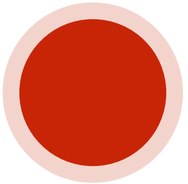
Server Core

호환성을 중시

완전한 .NET Framework 지원

대부분의 윈도우 백엔드 애플리케이션에 적합

4~5GiB 크기



Windows

GUI 애플리케이션 자동화에 적합

윈도우 OS 구성 요소의 대부분을 지원 (Internet Explorer, DirectX 포함)

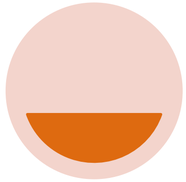
5GiB~ 크기

윈도 애플리케이션의 컨테이너화 (Cont.)

컨테이너화를 할 때 고려할 사항

- 윈도우 컨테이너는 GUI 지원이 안됩니다.
- 표준 입출력을 써야 하므로 NT 서비스는 콘솔로 바꿔야 합니다.
- 상황에 따라 다르지만 보통 아래의 기준을 사용합니다.
  - 기존 프로그램은 서버 코어 이미지로 만듭니다.
  - 윈도우 API 종속성이 약하다면 나노 서버 이미지로 만듭니다.
- 주요 언어, 프레임워크 프로젝트들이 윈도우 컨테이너 이미지도 제공 (참고: https://bit.ly/3ke2NUc)

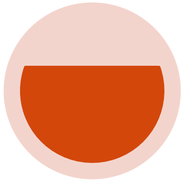
1.3 개발 환경을 구축하는 방법



Docker Toolbox

윈도우 10 1주년 업데이트 미만 대상

리눅스 컨테이너 전용 (VirtualBox)



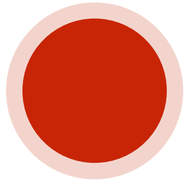
Docker Desktop

윈도우 10 최신 버전 대상

기본은 리눅스 컨테이너

윈도우 컨테이너 지원

개발 목적 전용!



Docker Enterprise

윈도우 서버 2016 이상 필요

윈도우 컨테이너 전용

쿠버네티스에 사용할 때는 윈도우 서버 2019 이상 필요

개발 환경을 구축하는 방법 (Cont.)

DEVIEW 2020

최신 버전의 Docker Desktop 설치

Docker Toolbox와 혼동하지 않도록
주의하세요!

Docker 공식 홈페이지를 통해 다운로드 가능

최신 버전의 윈도우 10 설치

단, Hyper-V 가상화를 사용할 수 있어야 합니다.

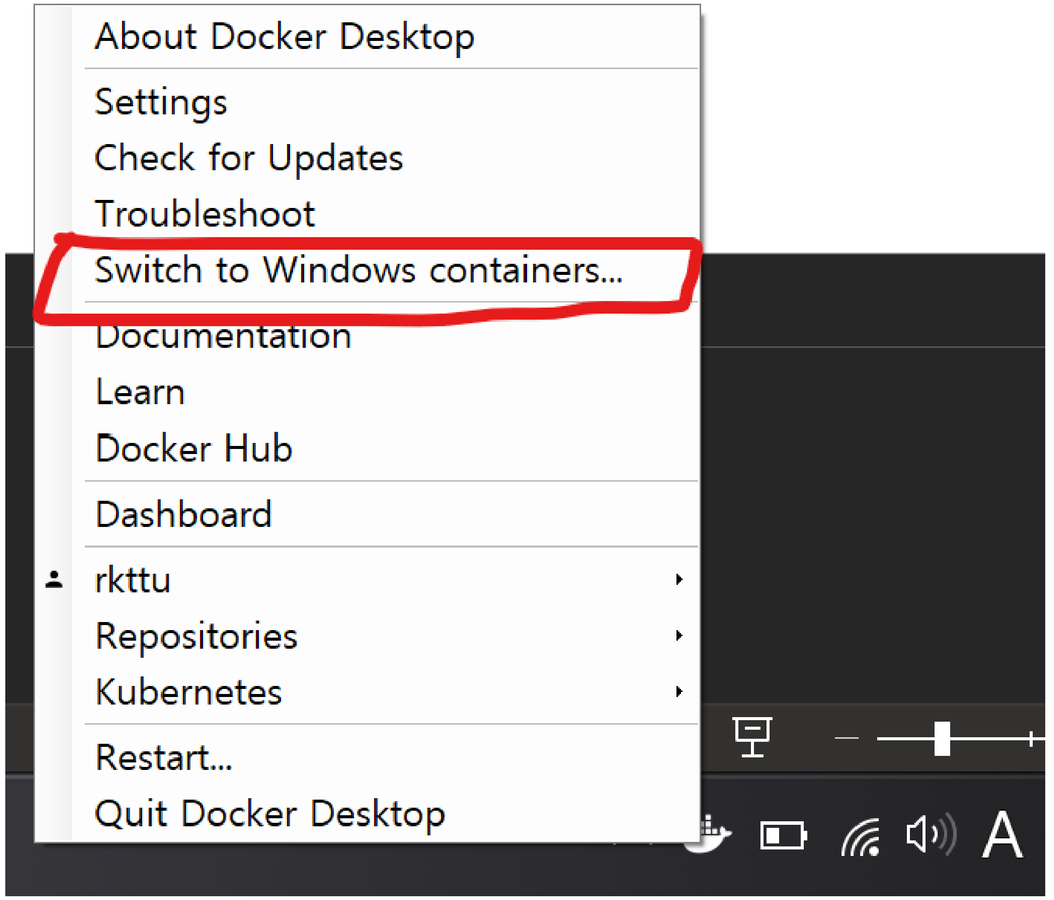
Intel Mac을 사용하시나요?
부트캠프가 좋습니다.

중요 - FileVault는 끄셔야 합니다!

개발 환경을 구축하는 방법 (Cont.)

기본은 리눅스 컨테이너 모드

- 트레이 아이콘 → 고래 모양 아이콘 → 오른쪽 클릭 → Switch to Windows containers… 선택
- Hyper-V 미 설치 시 자동으로 설치 후 시스템 다시 시작까지 진행

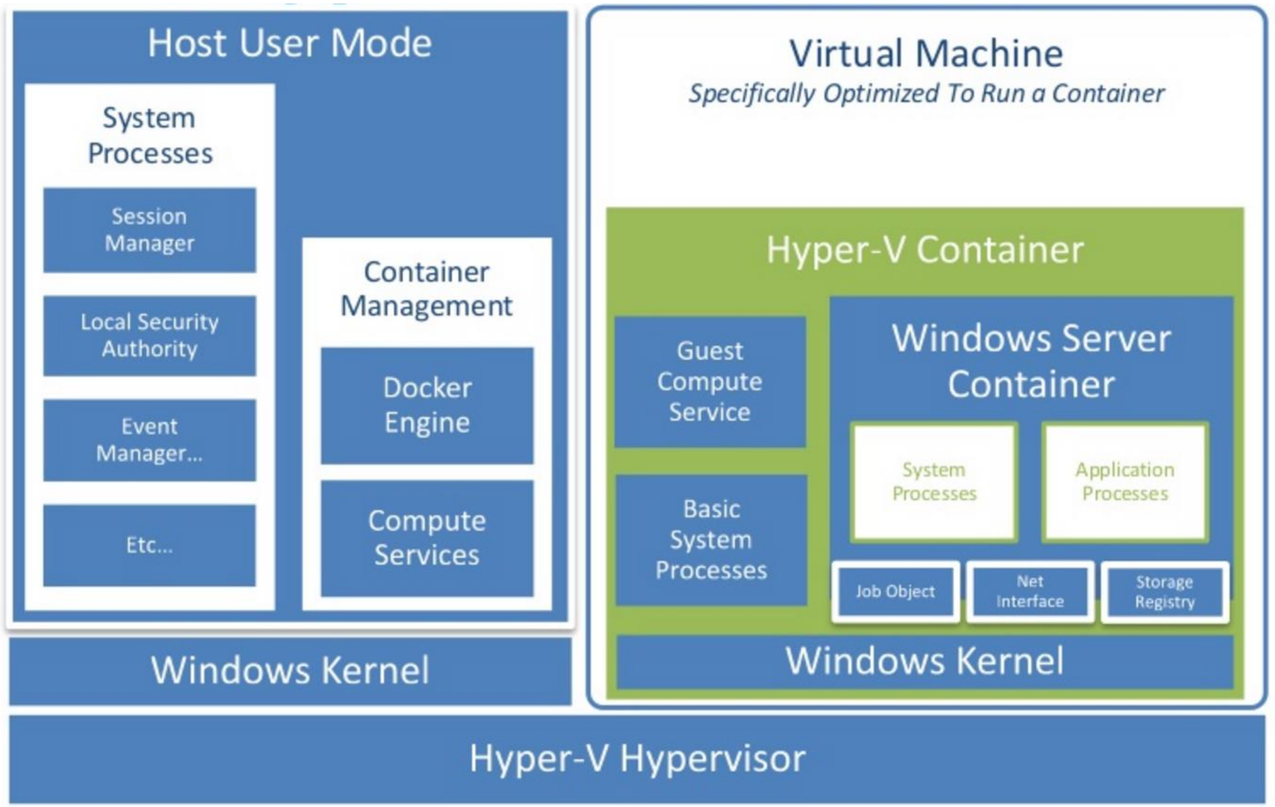


1.4. Hyper-V vs. 프로세스 격리

DEVIEW 2020

Hyper-V 격리 모드 기본 제공

- Hyper-V 컨테이너는 컨테이너 이미지 안에 포함된 윈도우 커널을 별도의 가상 실행 환경에서 불러들임
- 단, 전체 VM을 만들지 않고 컨테이너 실행에 필요한 부분만 가상화
- Guest Compute Service를 통해 호스트와 컨테이너 사이를 연계



Hyper-V vs. 프로세스 격리 (Cont.)

불가피한 Hyper-V 컨테이너 사용

- 호스트 OS 버전 > 컨테이너 이미지 버전
  - 이 경우 Hyper-V 컨테이너만 가능
- 호스트 OS 버전 == 컨테이너 이미지 버전
  - 자유롭게 선택 가능
  - 윈도우 쿠버네티스에 필요 조건
- 호스트 OS 버전 < 컨테이너 이미지 버전
  - 실행 불가

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

1.5 프로덕션 환경 선택, 라이선스 문제

윈도우 컨테이너를 프로덕션 서비스로 사용하신다면

- 프로세스 방식의 격리 모델 사용 시
  - 사용 개수 제한 없음
  - SKU 범위 제한 없음
- Hyper-V 격리 모델 사용 시
  - 개발 목적으로 사용할 때는 제한 없음
  - Standard Edition: 최대 2개 컨테이너
  - Datacenter Edition: 무제한
- 퍼블릭 클라우드 사용 시 라이선스 문제 고민을 하지 않아도 됨
  - 만약 기존 윈도 서버 라이선스가 있다면 라이선스 이동 가능 (비용 절약)

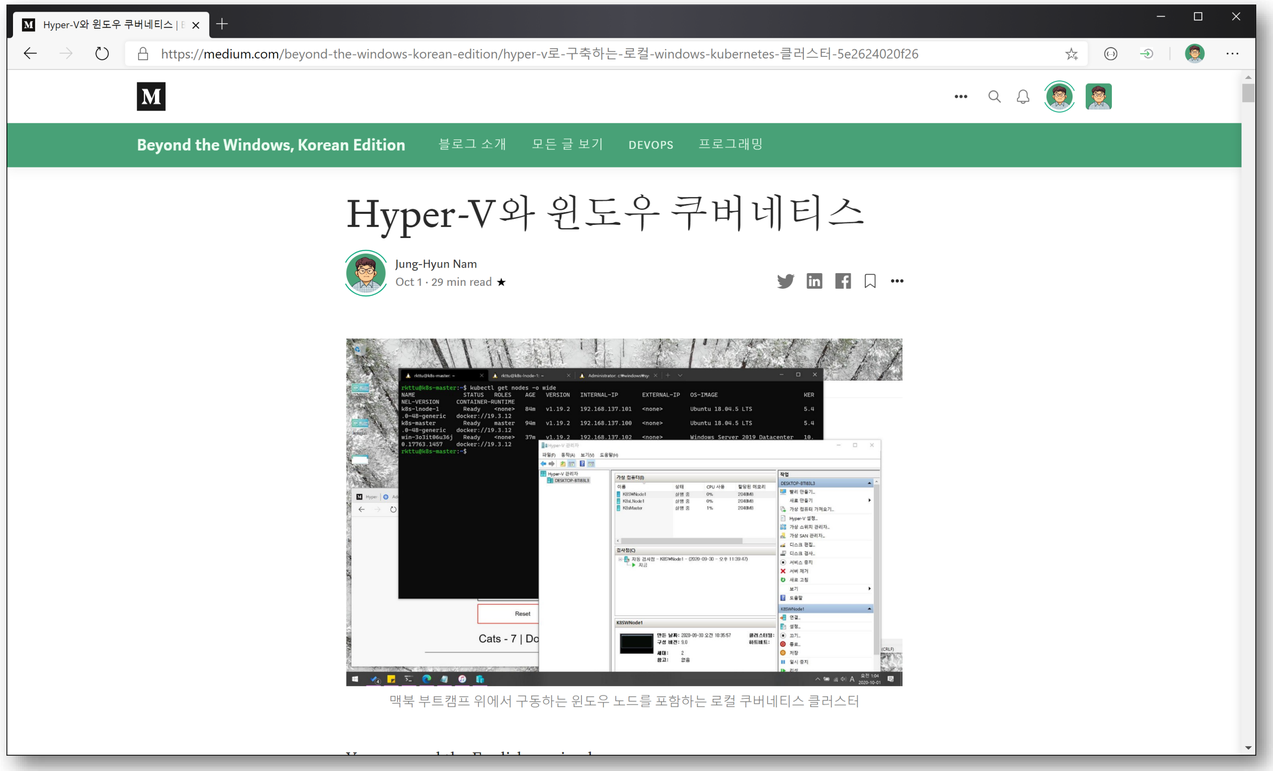
2.윈도우 쿠버네티스

계속하기 전에

내 노트북에 윈도우 쿠버네티스 클러스터 구축하기

이 자료를 잘 메모하고 나중에 활용해 보시면 시스템 이해에 도움이 됩니다.

https://bit.ly/3dn3vMd



맥북 부트캠프 위에서 구동하는 윈도우 노드를 포함하는 로컬 쿠버네티스 클러스터

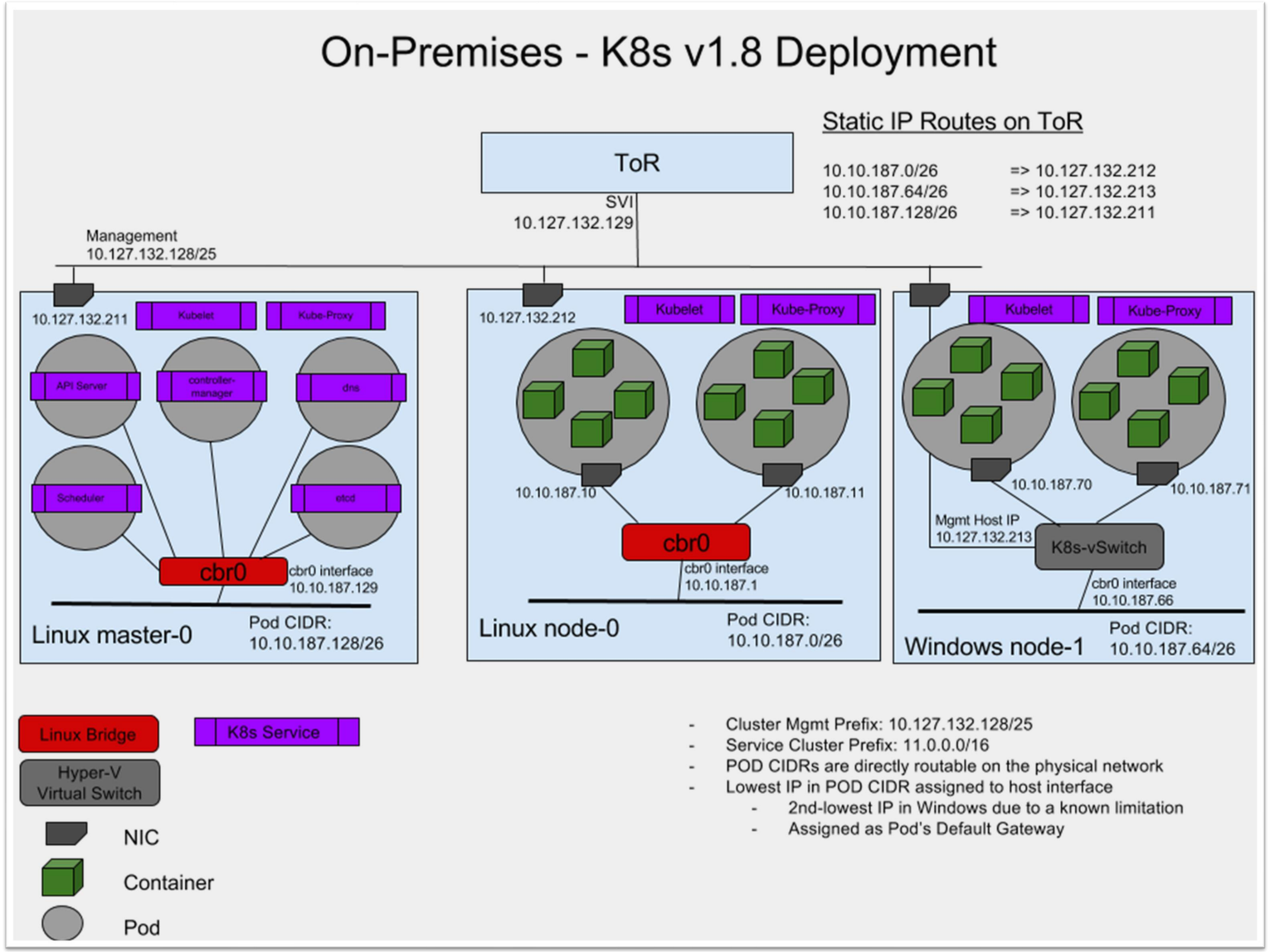
2.1 윈도우 쿠버네티스란?

DEVIEW 2020

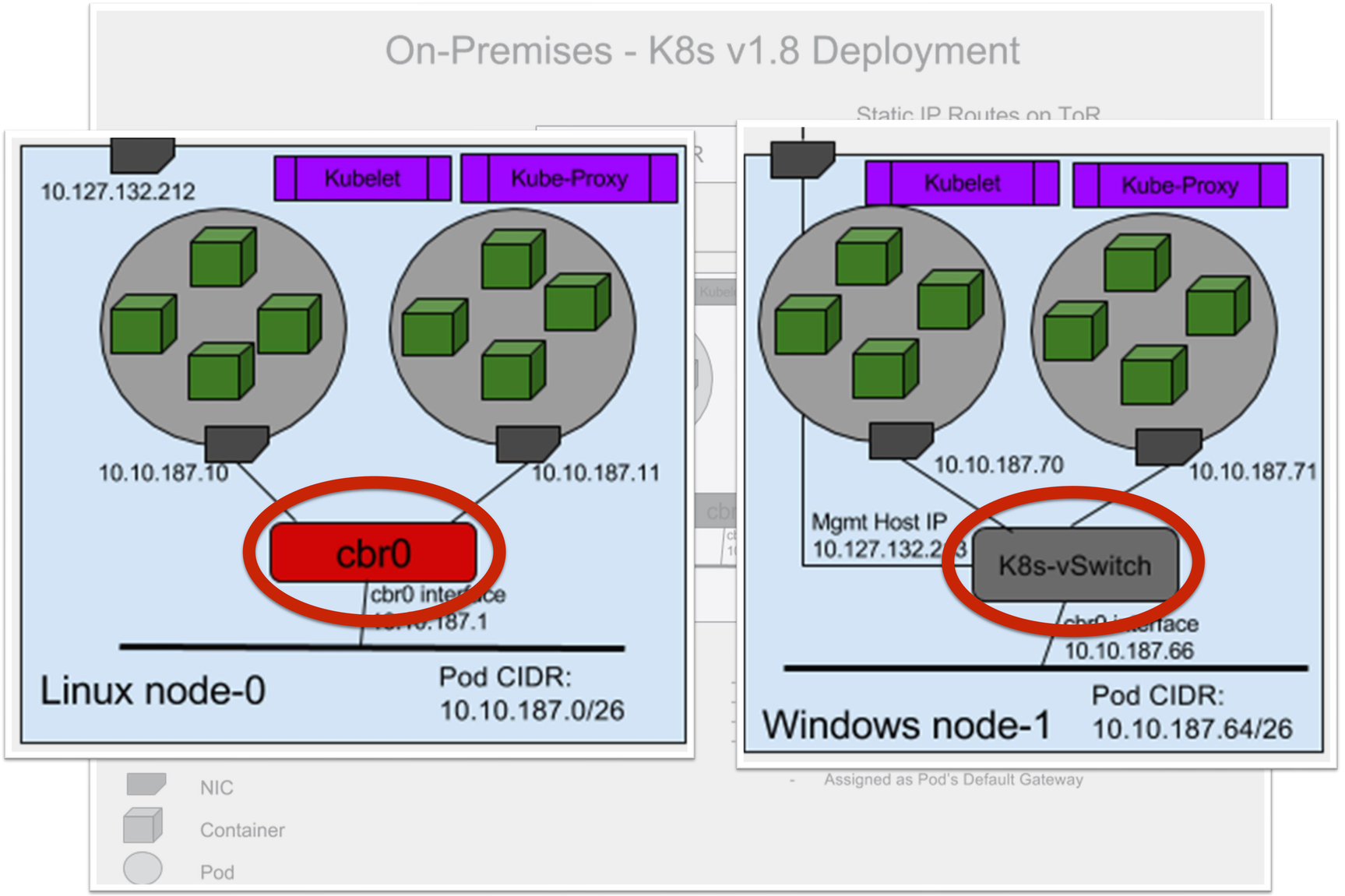
윈도우 쿠버네티스에 대한 간단한 소개

- 작업자 노드로 참가하는 컴퓨터가 윈도우 서버
  - 마스터 노드는 계속해서 리눅스를 OS로 사용함
- 윈도우 컨테이너를 쿠버네티스를 통해 배포, 관리할 수 있는 시스템
- 윈도우 서버 2016 이후의 주된 서버 OS 개발 방향
  - 쿠버네티스의 가상 네트워크를 지원할 수 있게 만듦
  - 리눅스 컨테이너와 동일한 기능 제공 (예: 단일 파일 마운트)

2.2 윈도우 쿠버네티스의 네트워킹

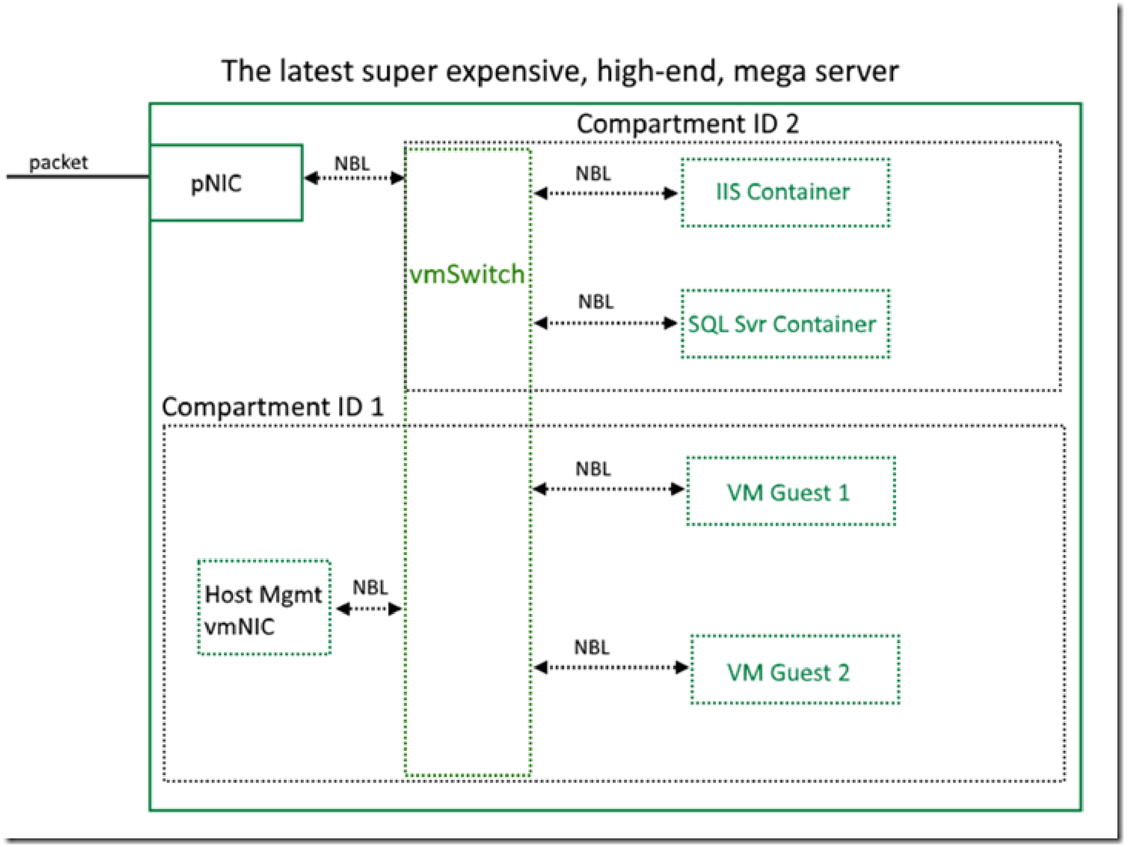


윈도우 쿠버네티스의 네트워킹 (Cont.)

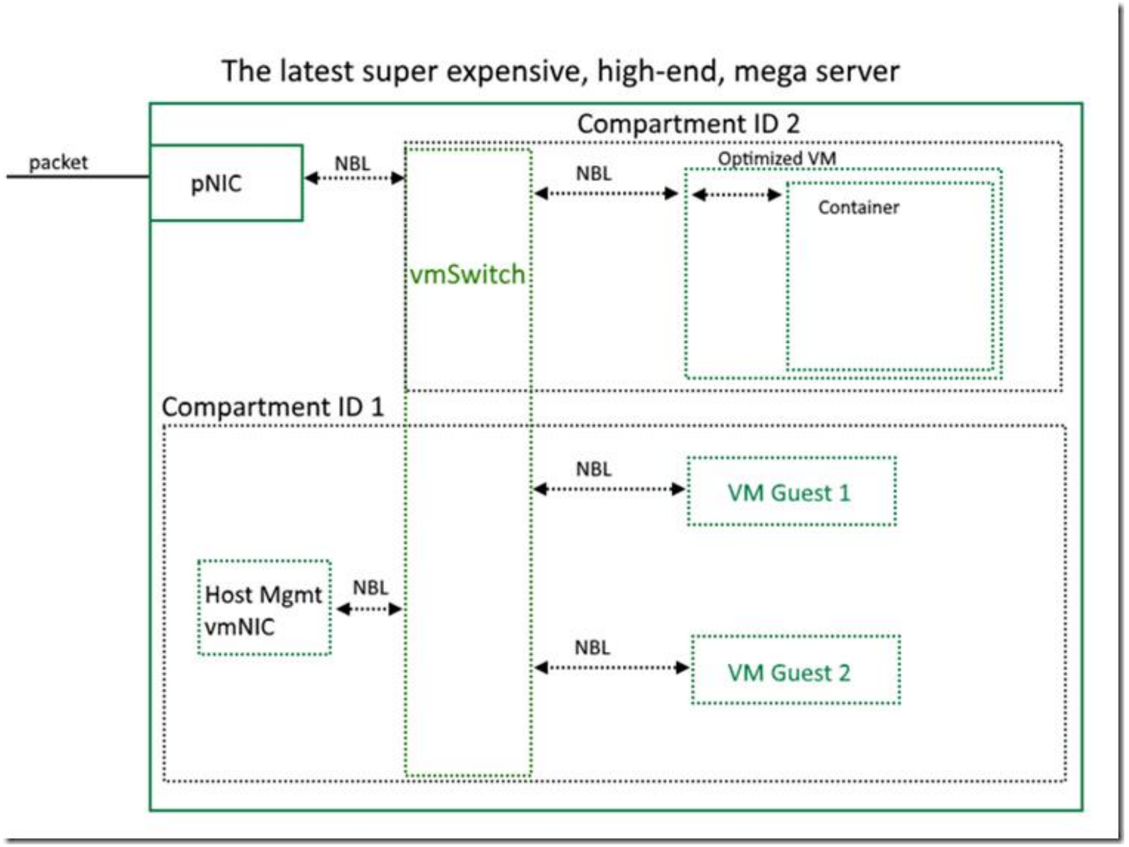


윈도우 쿠버네티스의 네트워킹 (Cont.)

프로세스 격리 방식



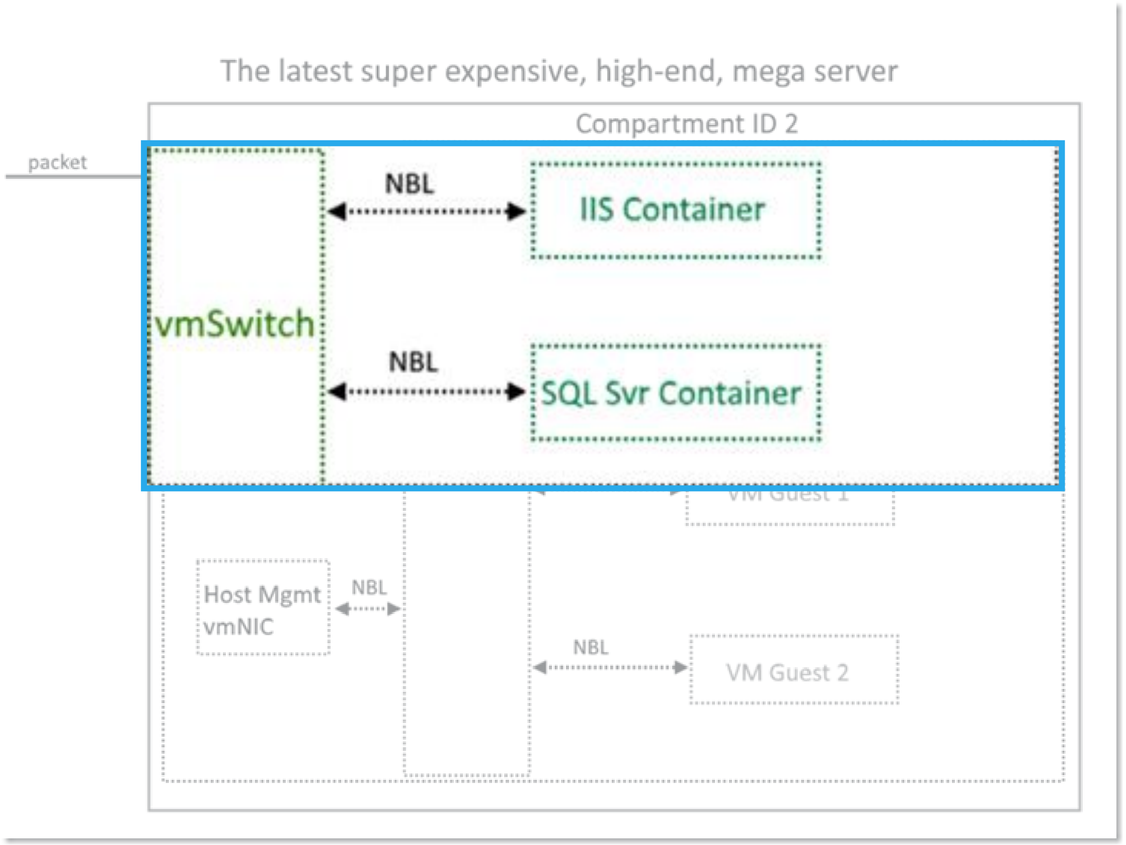
Hyper-V 격리 방식



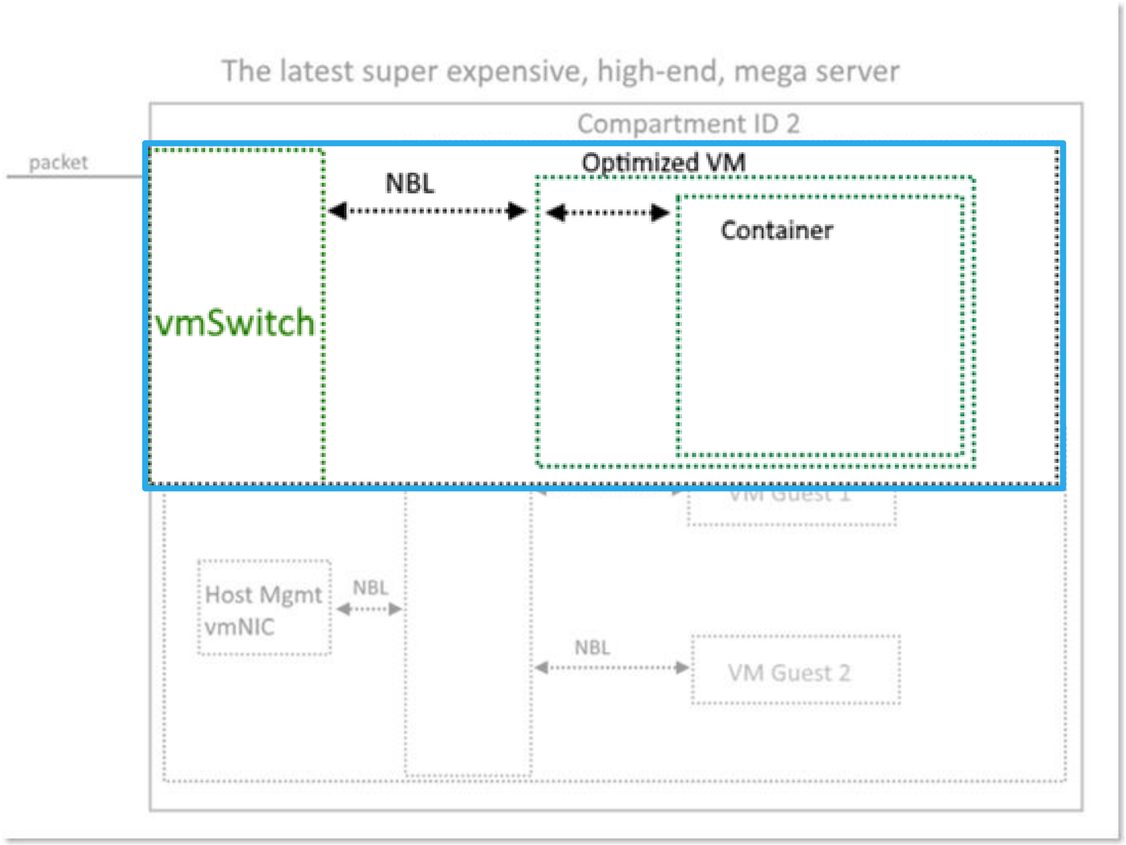
윈도우 쿠버네티스의 네트워킹 (Cont.)

DEVIEW 2020

프로세스 격리 방식



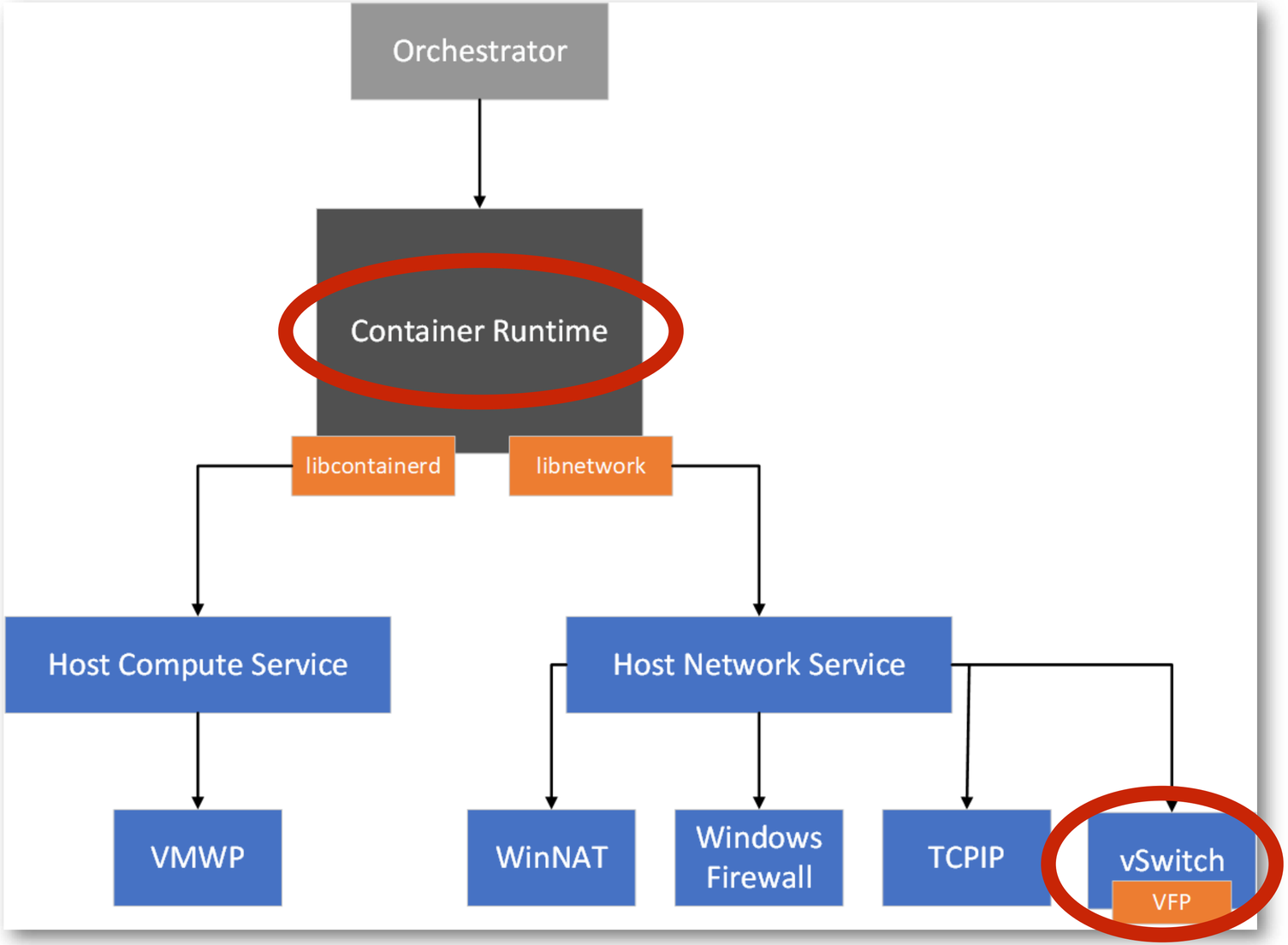
Hyper-V 격리 방식



윈도우 쿠버네티스의 네트워킹 (Cont.)

앞에서 살펴본 HCS 외에 HNS라는 것이 있습니다.

- HNS는 인터넷 공유, TCP/IP, 방화벽 기능과 함께 컨테이너 네트워킹을 제어합니다.



2.3 윈도 워커 노드의 부트스트래핑

윈도 워커 노드의 부트스트래핑 과정 요약

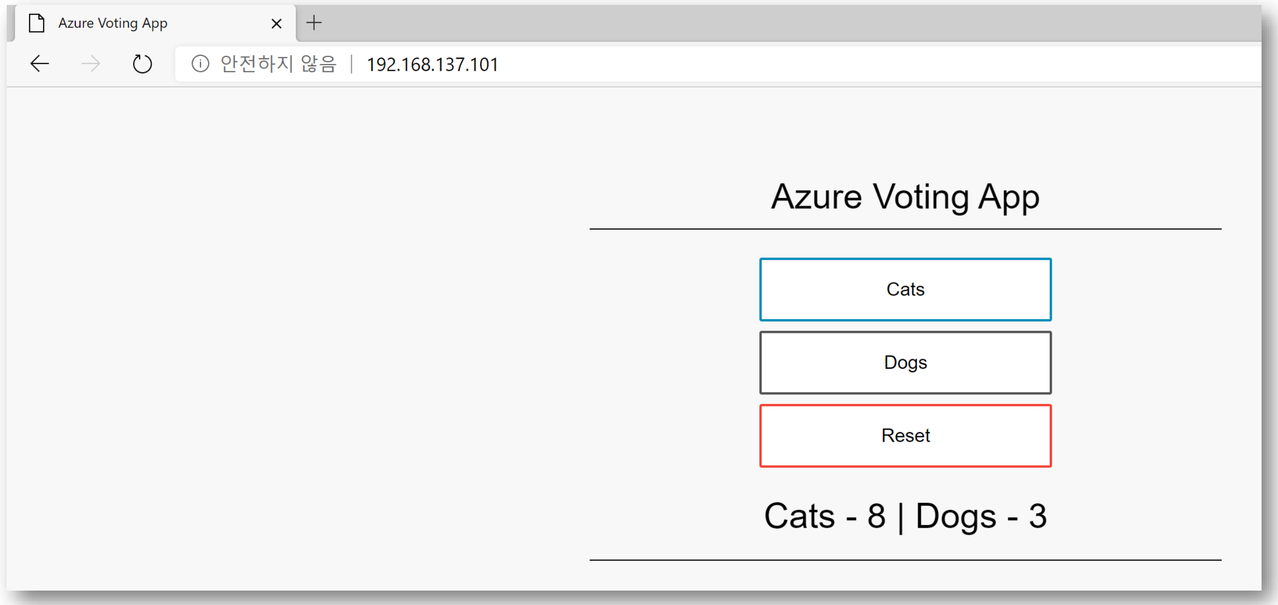
- Docker EE 및 kubelet.exe 사용
  - 추후 Docker EE 대신 윈도우용 conatinerd + CRI로 변경
- 네트워크 기능 구현은 kube-proxy.exe와 윈도우용 CNI가 담당
  - 앞에서 본 HNS를 이들이 제어하게 됨
  - 수동 배포 또는 Daemon Set 배포
  - 이 부분이 윈도우 쿠버네티스 설치에서 가장 까다로운 부분
  - 그 외 CNI 플러그인 수동 배포 또는 Daemon Set 배포
    - 윈도우용 Flannel 또는 윈도우용 Calico 사용 가능

2.4 실제 애플리케이션 배포하기

DEVIEW 2020

AzureVote 배포하기

- 프론트엔드, 내부 서비스, 백엔드, 로드밸런서로 구성
  - 프론트엔드: Python Django + 리눅스 컨테이너
  - 백엔드: 윈도우로 포팅된 Redis 3.0 + 윈도우 컨테이너



실제 애플리케이션 배포하기 (Cont.)

여러 OS와 아키텍처를 사용하는 노드에서의 배포 관리

- Node Selector 사용 필수
  - beta.kubernetes.io/os: windows
- Node Selector를 정확히 사용하지 않을 경우
  - 리눅스 노드가 윈도우 컨테이너 이미지를 Pull하거나 반대 상황 발생
- 하나 더, Taint의 사용
  - 윈도우 OS 버전 차이가 있으면 베이스 이미지도 같아야 함
  - 정확하게 배포하도록 만들려면 노드에 Taint 추가가 필수 (예: 1809, 1909)

실제 애플리케이션 배포하기 (Cont.)

아래 URL에서 Demo Video 시청이 가능합니다.

https://1drv.ms/v/s!Aj231qrFhlQxqcpqUks_eN1NeDIVdg?e=ua1QSx

2.5 프로덕션에서 사용하려면?

우리는 사람도 없고 시간도 없어요!

- 겉보기에 멀티 OS 클러스터는 매력적이긴 하지만…
  - 서로 다른 OS를 관리해야 하므로 당연히 관리 비용이 급상승!
- 그렇지만 멀티 OS 클러스터의 이점을 누리고 싶으시다면…
  - 시중의 3대 주요 퍼블릭 클라우드 업체 모두 윈도우 쿠버네티스를 제공합니다.

DEVIEW 2020

3.데브시스터즈와 윈도우 컨테이너

3.1 데브시스터즈와 윈도우 컨테이너

윈도우 컨테이너 도입을 통해 우리가 얻은 것

- 낙후된 윈도우 기반 게임 서버 인프라를 자동화
- 누구나 쉽게 띄우고 찍어낼 수 있는 게임 서비스
- 어떤 OS를 쓰던, 어떤 기술을 쓰던 진짜 모두 지원하는 인프라

지금은 게임 개발 과정에서 유용하게 사용하고 있습니다. ✌

데브시스터즈와 윈도우 컨테이너 (Cont.)

DEVIEW 2020

다시 한 번 감사 인사를 드립니다. 

- David Schott, Microsoft
- 김민규님
- 김지현님
- 이태화 수석님, VMWARE
- 인프라 셀
- 데이터 플랫폼 셀

데브시스터즈와 윈도우 컨테이너 (Cont.)

DEVIEW 2020

실제 개발 환경에 도입하기까지의 과정

- 애플리케이션 크래시 덤프 및 수집
- 임시 포트 소진 이슈
- 문제를 해결하느라 발생한 OS 버전 문제
- CI에 대한 고민과 솔루션
- Filebeat로 컨테이너 로그 수집
- Graceful Shutdown

3.2 애플리케이션 크래시 덤프 및 수집

DEVIEW 2020

크래시 덤프 수집의 자동화

- 앞서 이야기한대로 컨테이너마다 파일 시스템이 각각 존재
  - 만들어지는 크래시 덤프도 여기저기 흩어집니다.
- 이걸 모으기 위하여
  - 디렉터리 마운트로 호스트 디렉터리에 파일 모으기
  - 일정한 주기로 Object Storage에 파일을 업로드
  - Slack으로 메시지 보내기

3.2 애플리케이션 크래시 덤프 및 수집

DEVIEW 2020

How-to

- HKLM₩Software₩Microsoft₩Windows₩Windows Error Reporting₩LocalDumps
  - DumpFolder (Expandable String): 어느 위치에 덤프 파일들을 모을지?
  - DumpCount (DWORD): 덤프 파일을 최근 몇 개까지 남길지?
  - DumpType (DWORD): 만들 덤프 파일의 종류?

3.2 애플리케이션 크래시 덤프 및 수집

DEVIEW 2020

How-to

- 위의 설정에 따라 지정한 경로를 호스트에서 볼 수 있게 볼륨 마운트
  - 그 다음 Python 나노 서버 컨테이너로 덤프 자동 수집 + Slack 알림 발송
- Python 나노 서버 컨테이너는 아직 공식 버전이 없음
  - 커스텀 이미지 제작: github.com/rkttu/python-nanoserver

3.3 임시 포트 소진 이슈

윈도우 쿠버네티스 도입 초기에 겪었던 어려움

- 서비스 (HNS 입장에서는 로드밸런서) 당 64개 포트 사용
- 클러스터에 200개 이상의 서비스가 등록된 상태
- 사용 가능한 임시 포트의 개수는 평균 13000개 내외
- 사용 가능한 포트를 모두 소진하면 추가 연결이 불가능함

임시 포트 소진 이슈 (Cont.)

DEVIEW 2020

문제에 대한 임시 해결책으로…

- 서비스 개수 한도를 관리
- 동적 포트 범위를 더 넓게 설정 (netsh int ipv4 dynamicportrange)

그러나 궁극적으로는…

- 윈도우 서버 인사이더 프리뷰 빌드 18945 이상 사용
- Kube-proxy의 WinDSR 피처 게이트 사용

임시 포트 소진 이슈 (Cont.)

DEVIEW 2020

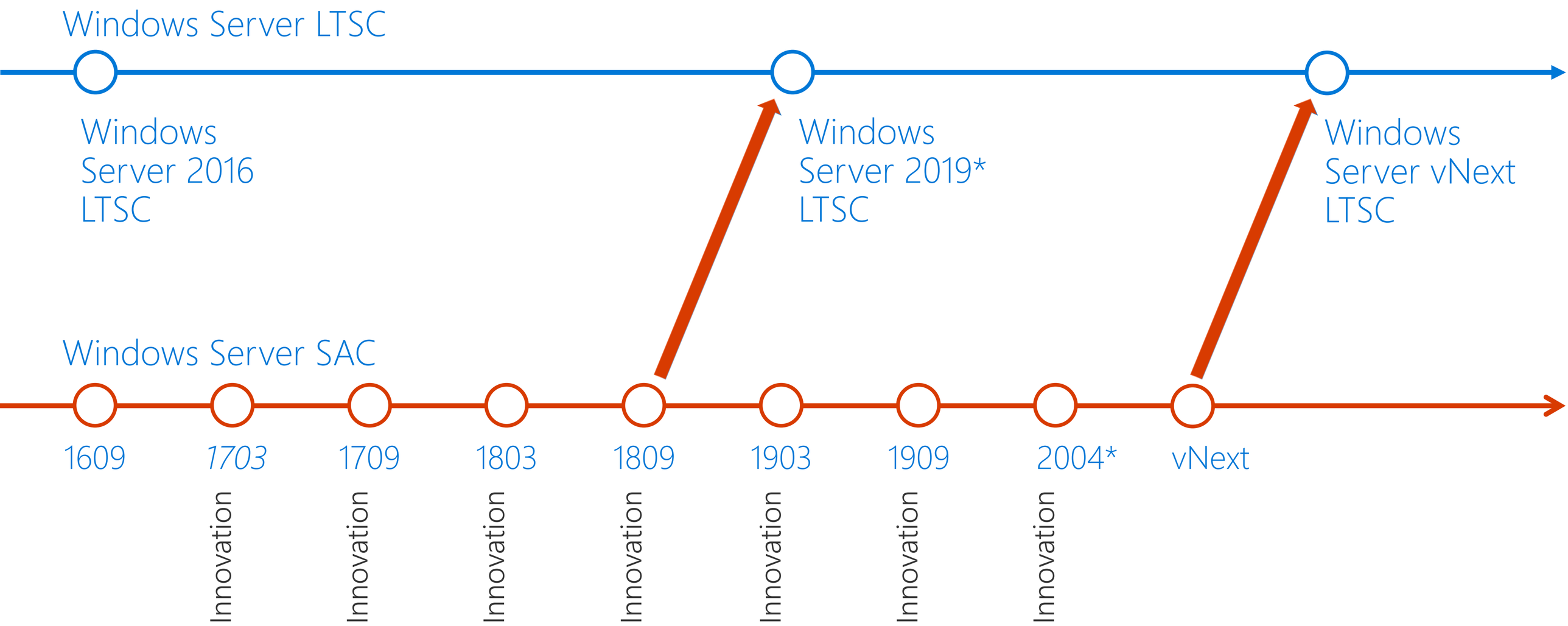
문제 해결을 위하여 아래의 방법을 택했습니다.

- kube-proxy NT 서비스 시작 인자 변경
  → 파워셸로 패치 스크립트를 만들어 초기화 스크립트에 포함
- 윈도우 서버 1903으로 서버 OS 업그레이드

```
249 +
250 +   additional_userdata_windows = <<EOFTF
251 + ${file("patch.ps1")}
252 +
253 + $InstanceId = Fetch-AWSInstanceId
254 +
255 + $TargetPath = 'HKLM:\SYSTEM\CurrentControlSet\Services\kubelet\'
256 + $ArgumentString = (Get-ItemProperty -Path $TargetPath).ImagePath
257 + $ArgumentString = Compose-EKSKubeletArg -ArgumentString $ArgumentString -AwsInstanceId $InstanceId
258 + Set-ItemProperty -Path $TargetPath -Name 'ImagePath' -Value $ArgumentString
259 + net.exe stop kubelet
260 + net.exe start kubelet
261 +
262 + $TargetPath = 'HKLM:\SYSTEM\CurrentControlSet\Services\kube-proxy\'
263 + $ArgumentString = (Get-ItemProperty -Path $TargetPath).ImagePath
264 + $ArgumentString = Compose-EKSKubeProxyArg -ArgumentString $ArgumentString -WinOverlayEnabled $true -WinDSREnabled $true
265 + Set-ItemProperty -Path $TargetPath -Name 'ImagePath' -Value $ArgumentString
266 + net.exe stop kube-proxy
267 + net.exe start kube-proxy
```

3.4 문제를 해결하느라 발생한 OS 버전 문제

DEVIEW 2020



문제를 해결하느라 발생한 OS 버전 문제 (Cont.)

LTSC vs. SAC

- 윈도우 서버 2016부터는 버전 체계가 둘로 나뉩니다.
  - 반기 (6개월)에 한 번 업데이트되는 SAC 버전
  - 약 2~3년에 한 번 업데이트되는 LTSC 버전
- 도커 스웜 등과는 달리 쿠버네티스의 윈도우 지원은 계속 향상 중
  - 원하는 기능이 현재 버전의 OS에 없을 수 있음
  - LTSC 버전이 아닌 SAC 버전을 사용하기 까다로울 수 있음
  - 최신 기능을 사용하려면 심도있는 클러스터 구축과 관리가 필요

3.5 CI에 대한 고민과 솔루션

DEVIEW 2020

임시 포트 소진 이슈를 해결한 후 또 다시 만난 문제

- 우리는 SAC 버전의 윈도우 서버를 쓰는데…
  - Github이나 CircleCI는 LTSC 버전만 CI를 제공함
- 이 문제를 해결하기 위하여 Github Action Runner를 도입

CI에 대한 고민과 솔루션 (Cont.)

Github Runner 설치와 구성

- 우리 팀은 주로 윈도우 컨테이너 빌드를 수행
  - Docker EE외에 다른 도구는 설치하지 않았음
  - 상황에 따라 Visual Studio Build Tools도 추가 가능
- 별도의 Github Runner 전용 윈도우 계정 생성
  - 윈도우 서버의 강력한 암호 기준 충족을 위해 (사람이 쓸 비밀 번호 X)
    - System.Web.Security.Membership의 GeneratePassword 사용
  - Github Runner 설치 시 만든 윈도우 계정으로 등록
    - 인스턴스를 재부팅하더라도 항상 사용 가능한 상태 유지

3.6 Filebeat로 컨테이너 로그 수집

비슷한 듯 다른 Docker의 실행 환경

- Docker 컨테이너 로그를 Filebeat 에이전트가 수집
  - Docker 관련 메타데이터를 Docker로부터 가져와 추가해서 전송
- 리눅스와는 달리 우선 Filebeat로 Docker EE의 로그 수집부터가 허들!
  - 패키지 매니저가 아닌 ZIP 파일을 통한 설치 과정 자동화
- 어떻게 Docker Engine과 통신할 수 있게 만들까?
  - 리눅스에서는 Docker 소켓을 사용했지만 윈도우에서는?
  - 명명된 파이프가 그 역할을 대신함 (₩₩.₩pipe₩docker_engine)
  - 다행히 명명된 파이프를 호스트와 컨테이너가 공유 가능

Filebeat로 컨테이너 로그 수집 (Cont.)

Docker 컨테이너 실행 예시:

```
docker run --rm -it ^
  -l sample_label=hello-world ^
  -l cloud.devsisters.logging.enabled=true ^
  -v \\.\pipe\docker_engine:\\.\pipe\docker_engine ^
  -v "%ProgramData%\docker\containers":"C:\ProgramData\docker\containers" ^
  -v "%~dp0config":"C:\ProgramData\filebeat\config" ^
  filebeat-win32:servercore-ltsc2019 -- ^
  -c "C:\ProgramData\filebeat\config\filebeat.yml"
```

Filebeat로 컨테이너 로그 수집 (Cont.)

DEVIEW 2020

쿠버네티스로 이것을 띄우려면 어떻게 해야 할까?

- 앞에서 이야기한 대로 노드 셀렉터를 사용하여 윈도 노드에만 적용되야!
- Filebeat.yml 파일의 구성?

```
data:
  filebeat.yml: |-
    filebeat.inputs:
      - type: container
        paths:
          - 'C:\ProgramData\Docker\containers\*\*.log'
        processors:
          - add_kubernetes_metadata:
              in_cluster: true
              include_annotations: true
          - add_docker_metadata:
              host: 'npipe:////./pipe/docker_engine'
```

3.7 Graceful Shutdown

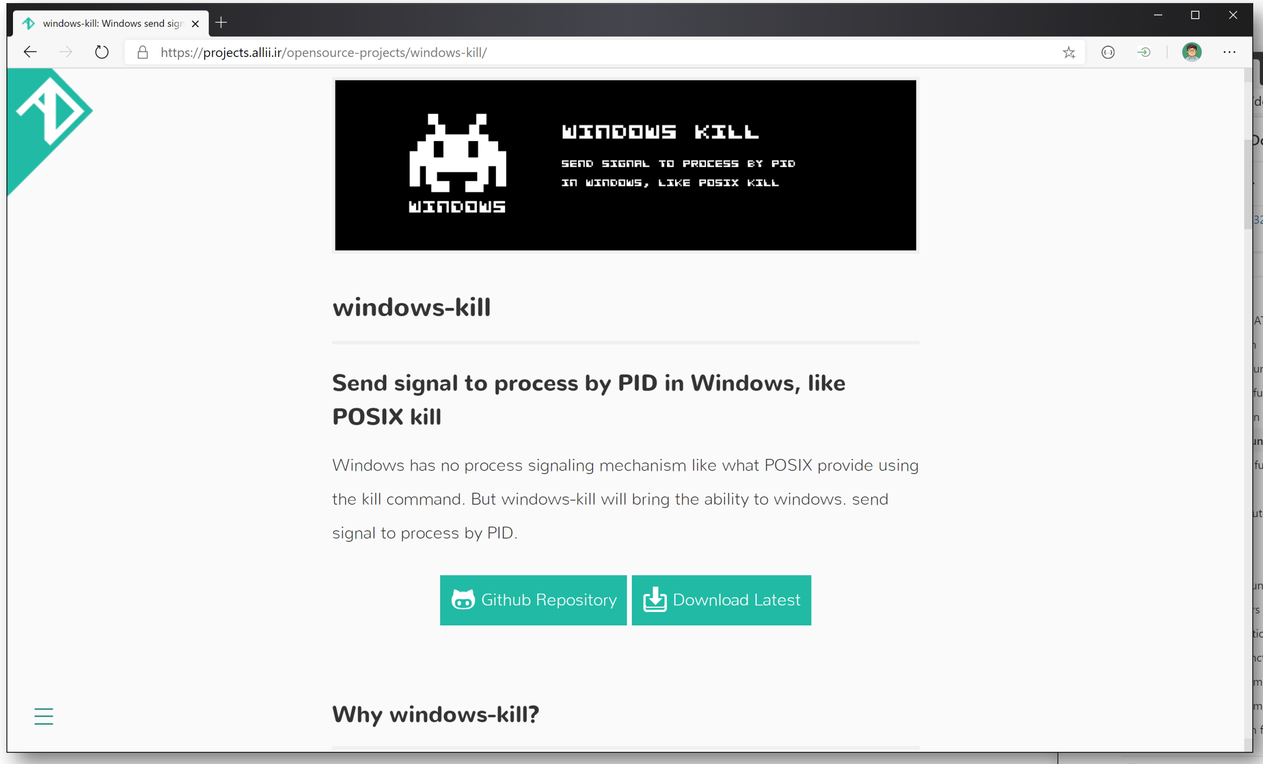
실시간 – 또는 – 연결 지향의 게임 서버를 다루면서 고려해야 할 또 다른 요소로 점진적인 서버 종료에 대한 것이 있습니다.

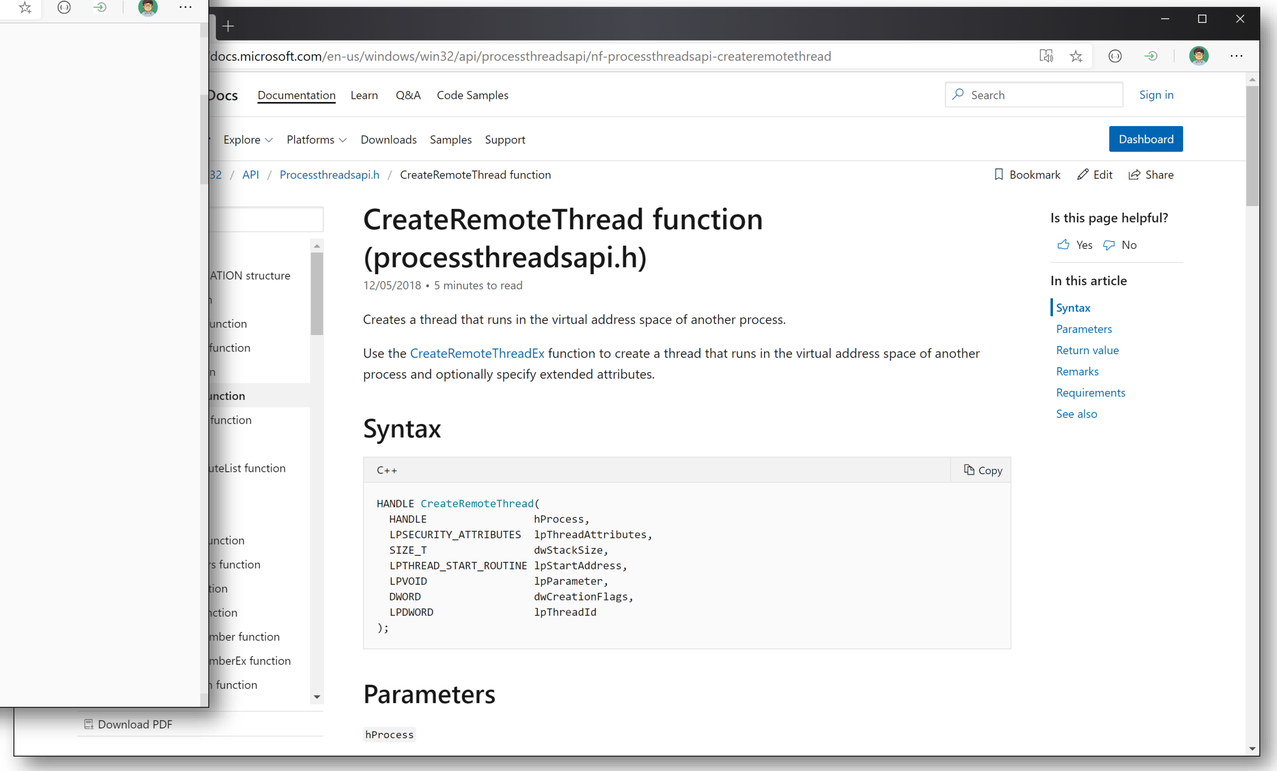
리눅스의 경우 시그널을 통해 즉시 종료만이 아니라 종료 준비 등을 요청하는 목적으로 이를 구현할 수 있습니다.

하지만 윈도우 컨테이너에서는 역시 쉽지 않은 문제입니다.

Graceful Shutdown (Cont.)

많은 해결 방법을 찾던 중에 CreateRemoteThread API를 활용하는 windows-kill 이라는 도구를 찾게 되었습니다.





Graceful Shutdown (Cont.)

VM 종료 훅 수신

일정 시간 동안 종료를 유예

컨테이너로 windows-kill 프로그램 복사

종료 과정 진행

연결된 메인 프로세스 ID 확인 후 windows-kill 실행

유예 기간 내에 서버가 종료됨을 클라이언트에 통지

유예 기간이 지나면 안전하게 접속 종료

서버 종료

이후 종료 과정은
일반적인 OS 및 VM 종료
절차를 따라 진행

4. 윈도우 컨테이너 기술의 미래

4.1 윈도우 서버와 containerd

아직까지 윈도우 쿠버네티스는 구 버전의 HCS를 기반으로 하는 Docker Enterprise Edition을 직접 사용하고 있습니다.

이것을 최신 버전의 HCS를 사용하는 윈도우용 containerd와 CRI로 이동하는 태스크 아이템이 진행 중입니다.

관련 태스크 아이템: https://bit.ly/2IfdbNh

윈도우 서버와 containerd (Cont.)

프로세스 격리 방식에 대한 절반의 믿음

- 서버에 직접 프로그램을 설치해서 실행하는 것보단 훨씬 안전합니다.
  - 하지만 보안 침해 사고는 늘 예상 못하는 곳에서 오는 법
- 좀 더 오버헤드가 발생하더라도 Hyper-V 컨테이너를 쓰면 좋은점
  - OS 버전과 컨테이너 이미지 버전이 무조건 일치하지 않아도 됩니다.
  - 보안 침해 사고가 발생할 확률을 좀 더 낮춰줍니다.
- 그러나…
  - 베어 메탈 서버나 중첩 가상화가 가능한 환경이 필요합니다.
  - 그 외의 경우에는 프로세스 격리 방식이 합리적입니다.

4.2 WSL v2와 LCOW

윈도우에서 리눅스 컨테이너 돌리기?

- 한 때 Docker EE에서 리눅스 컨테이너까지 지원하려고도 했습니다만
  - 현재는 모두가 잘 아는 WSL v2 + Docker로 대체되었습니다.
  - 서버 환경에서는 윈도우 노드와 리눅스 노드를 잘 구분해주세요.

4.3 이런 저런 동향들

윈도우 컨테이너는 계속 개발 중입니다.
https://bit.ly/2GKnCZ7

윈도우에도 드디어 Envoy 알파 버전 출시!
https://bit.ly/3nIL4q4

윈도우용 Calico CNI가 오픈 소스로 공개되었습니다.
https://bit.ly/372SRZU

IPv4, IPv6 듀얼 스택을 사용할 수 있습니다.
https://bit.ly/376WbDc

엔드포인트 슬라이스를 사용할 수 있습니다.
https://bit.ly/30WZybZ

We’re Hiring!

careers.devsisters.com

linkedin.com/company/devsisters

고맙습니다!