DEVSISTERS
한국 Azure 사용자 그룹
남정현

Kubernetes에서 Windows Application 호스팅하기

Agenda

- Case Study: DEVSISTERS
- Windows Kubernetes에 대하여
- HCS와 HNS에 대하여
- Windows 워커 노드 구성 분석
- Windows 컨테이너 실행해보기
- 약간의 추가 정보

DEVSISTERS의 사례

리눅스 워크로드와 윈도 워크로드를 하나의 클러스터에서.

Kubernetes의 고가용성 전략을 Windows에서 그대로 사용

Windows 만의 장점인 고성능 I/O Completion Port 지원

기존의 Legacy Application을 클러스터 환경으로 손쉽게 확장

DEVSISTERS의 사례 (Cont.)

하이브리드 K8S 클러스터 구축

- Windows와 Linux Pod이 공존하며 Cross Communication 전개

쿠키워즈 개발/QA 환경

- 게임 서버: 리눅스
- 운영 도구: 윈도 서버 1803
- 클러스터 구축: KOPS + AWS

두 번의 PoC 수행

- 초기 – Windows Server 2016 AMI + Transparent Network
- 현재 – Windows Server 1803 AMI + L2 Bridge + WINCNI

Windows Kubernetes

- 일반적인 Kubernetes와 다르지 않음
- Kubernetes의 고가용성 전략을 Windows에서 그대로 사용
- Windows 만의 장점인 고성능 I/O Completion Port 지원
- 기존의 Legacy Application을 클러스터 환경으로 손쉽게 확장

Kubernetes Windows SIG

- https://github.com/kubernetes/community/tree/master/sig-windows
- Windows 버전의 Kubernetes에 대한 활동을 전문으로 함
- 2018년 6월 현재 두 명의 Chair가 활동 중. (@michmike, @patriklang)

Windows Kubernetes History

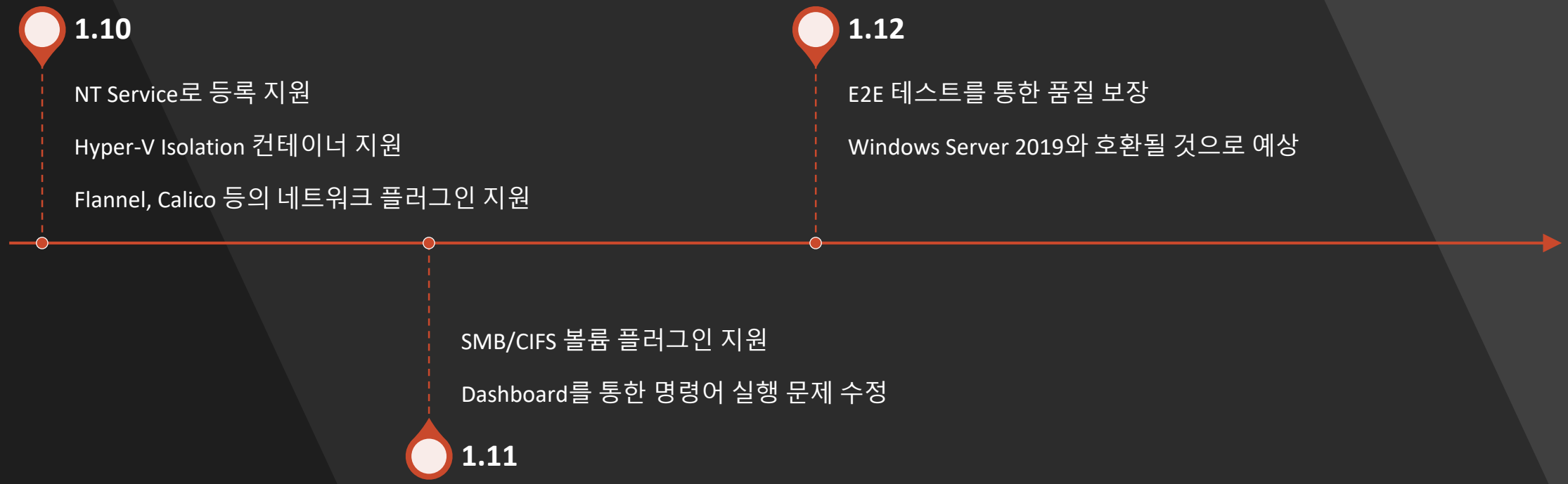
Kubernetes 1.5

- Alpha State
- Transparent Network
- Windows Server 2016
- 몇 가지 도드라지는 제약 사항 (Outbound 불가)

Kubernetes 1.9

- Beta State
- L2 Bridge
- Windows Container Network Interface
- Windows Server 1709 이상 필요 (Azure, AWS에서 사용 가능)

Road to Production



Windows Kubernetes 설치 과정

1

Windows Server 1803 VM 또는 베어메탈 준비

2

Node를 시작하기 전에 kubelet으로 노드를 먼저 등록하여 Pod CIDR 확보

3

HNS에 새로운 L2 Bridge HNS Network 생성, Pod Gateway 어댑터 생성 및 부착

4

Kubelet, Kubeproxy 설정 후 기동

5

Pod 간 수동 Routing Table 등록

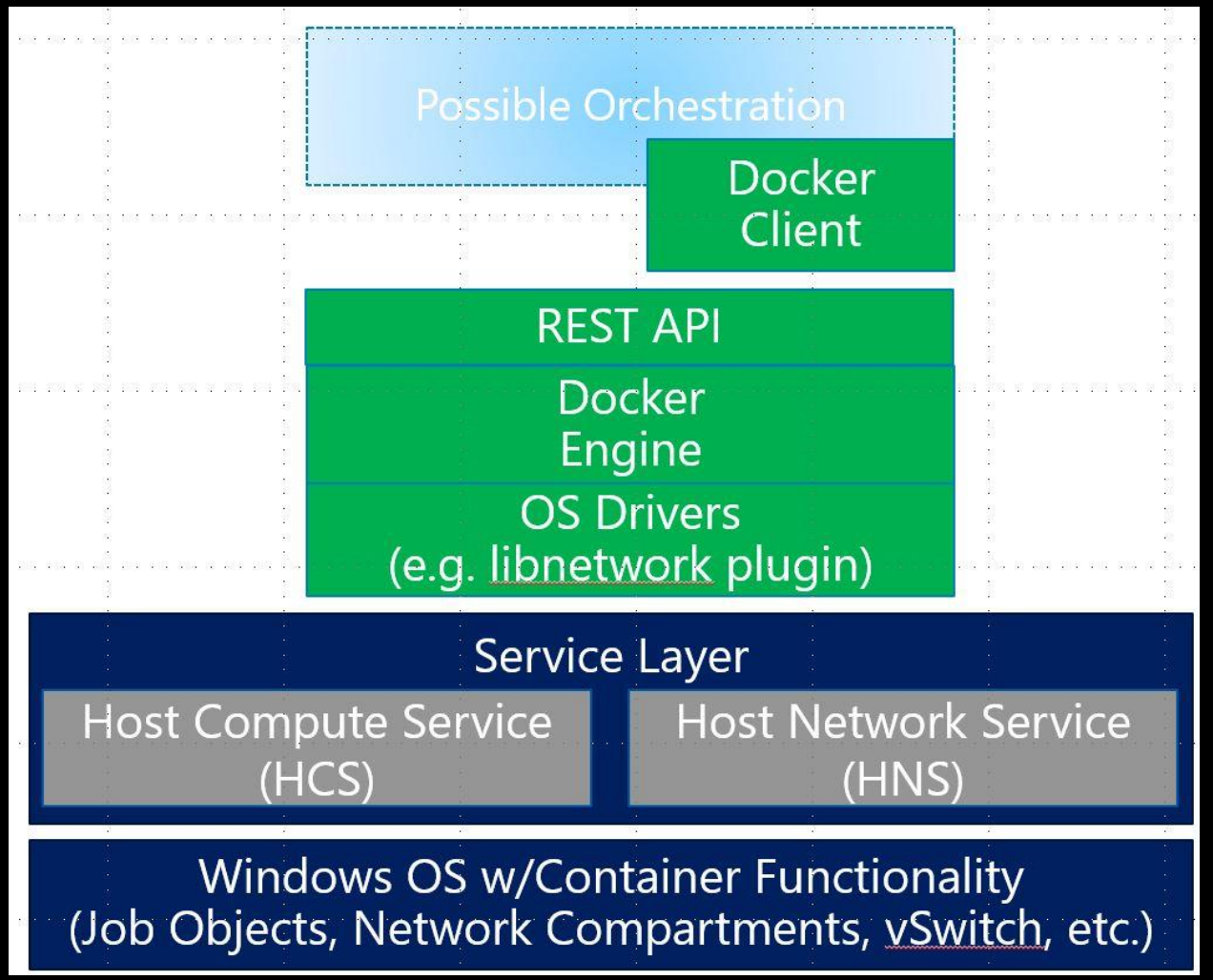
Windows 컨테이너의 버전 선택

- 커널 구조의 차이로 도입
  - Process 방식, Hyper-V 방식
- Windows 10에서 Windows Container를 실행 시
  - 클라이언트 커널 ≠ 서버 커널
  - Hyper-V Isolation 필요
- 버전 간 호환성에 대하여
  - https://bit.ly/2JSTo5A

Host > Container V	Win Server 2016	Win 10 Creators Update 1703	Win Server 1709	Win 10 Fall Creators Update 1709	Win Server 1803	Win 10 2018 April Update 1803
Win Server 2016	Process Hyper-V	Hyper-V	Hyper-V	Hyper-V	Hyper-V	Hyper-V
Win Server 1709	사용 불가	사용 불가	Process Hyper-V	Hyper-V	Hyper-V	Hyper-V
Win Server 1803	사용 불가	사용 불가	사용 불가	사용 불가	Process Hyper-V	Hyper-V

Windows Kubernetes의 핵심

- HCS와 HNS가 핵심
- Kubernetes 입장에서는 Docker REST API와 통신
- Docker는 HCS와 HNS와 커뮤니케이션 진행
- Windows의 경우 추후 CRI (Container Runtime Interface)가 도입될 예정



Host Compute Service

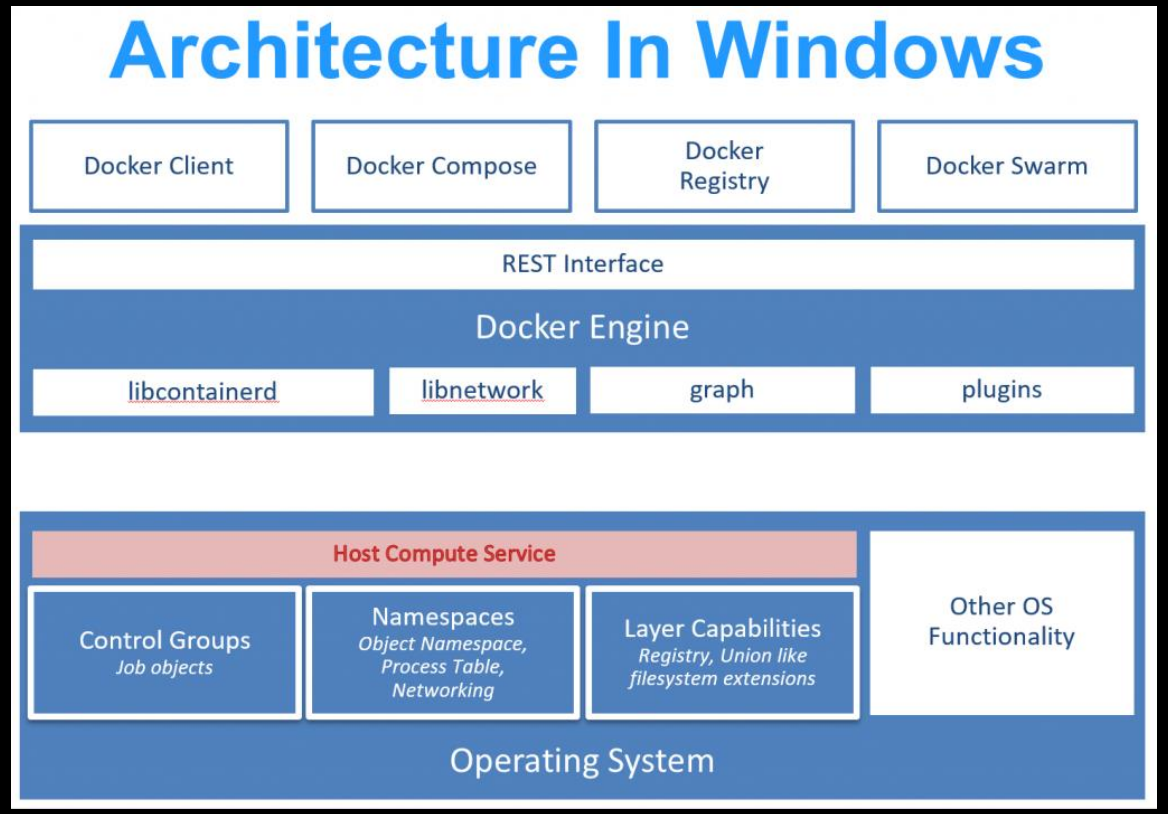
커널 수준의 가상화

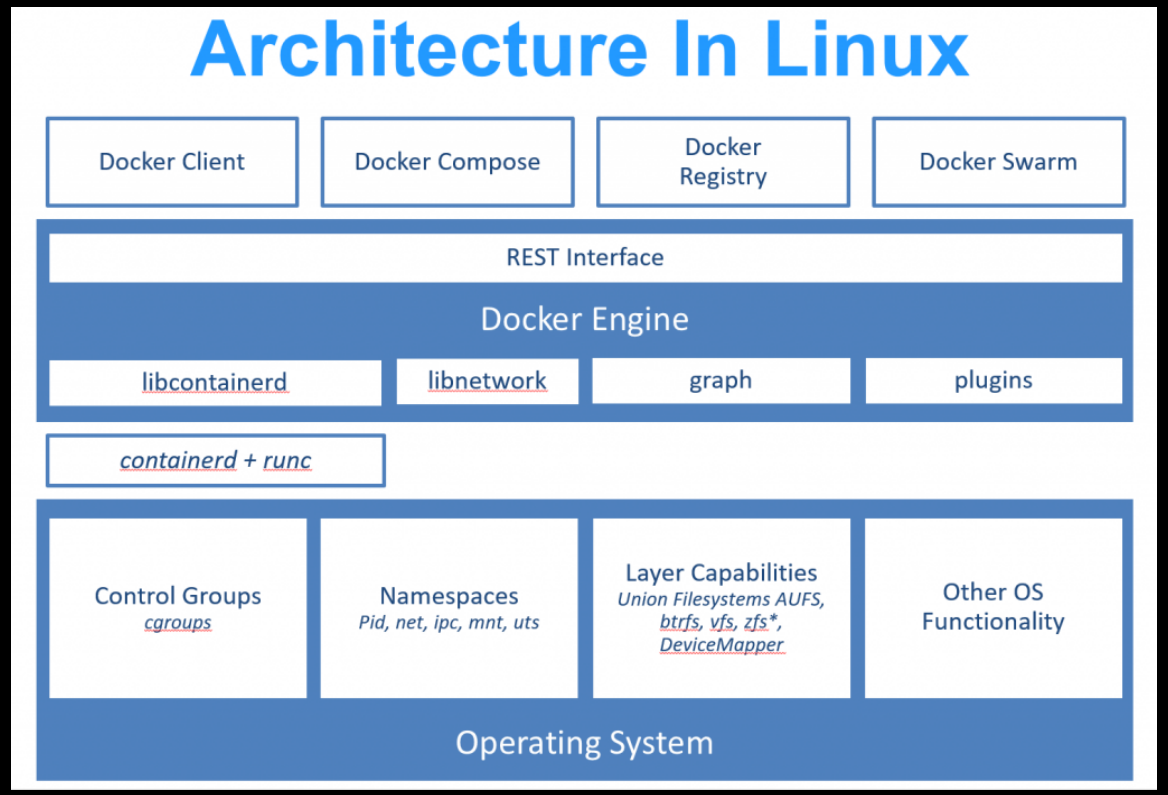
- 컨테이너 내부의 OS 버전과 호스트 OS 버전이 반드시 일치해야 함
- Windows 10 클라이언트 PC에서 서버 커널은 이 방식으로 호스팅이 불가하며, 오로지 Hyper-V Isolation만 지원

Hyper-V Isolation

- 가장 많이 오해를 받는 Windows Docker 확장 기능
- 전체가 격리된 VM은 아니기 때문에 호스트 OS 버전보다 높은 버전의 컨테이너 OS는 실행 불가
- https://bit.ly/2JSTo5A

https://blogs.technet.microsoft.com/virtualization/2017/01/27/introducing-the-host-compute-service-hcs/





Host Compute Service (Cont.)

Host Compute Service (Cont.)

- Linux 커널의 일부 컴포넌트에 대응되는 기능을 하나의 컴포넌트로 통합 제공
  - Control Group, Namespace
  - Layer Capabilities: Union File System + Registry
- Windows Kubernetes에 공헌할 목적으로 Microsoft가 직접 Go Lang으로 HCS Shim도 제공함
  - https://github.com/Microsoft/hcsshim
- .NET Framework 버전의 HCS 제어 프로그램 소스 코드도 제공함
  - https://github.com/Microsoft/dotnet-computevirtualization

Host Network Service

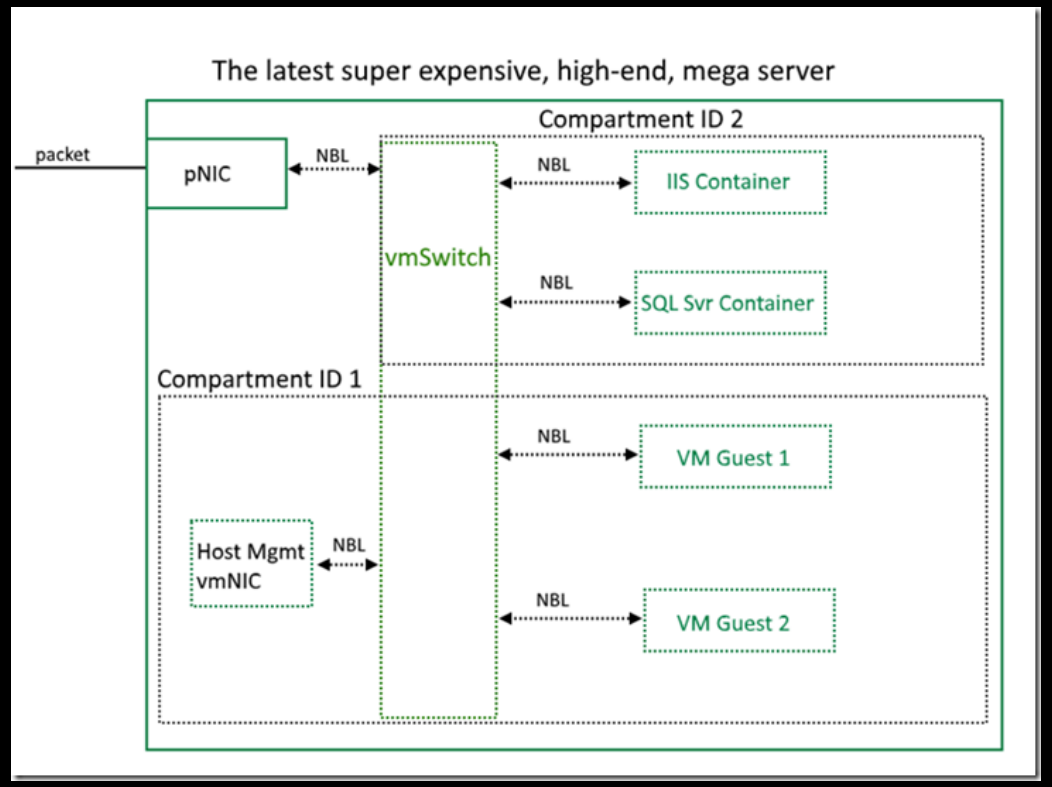
- 각종 네트워크 설정 제어 가능
  - https://github.com/Microsoft/SDN
  - /Kubernetes/Windows/HNS.psm1
  - VmCompute.dll의 HNSCall 메서드 사용
  - REST API 방식으로 호출 (Method + Resource Path + JSON Payload)
- HNS를 이용하여 주로 제어하는 부분
  - 네트워크: 내부 네트워크
  - 엔드포인트: 컨테이너들이 사용하는 가상 어댑터
  - 정책: 다른 노드 (리눅스, 윈도)에서 실행되는 Pod의 네트워크 연결 정보

Host Network Service (Cont.)

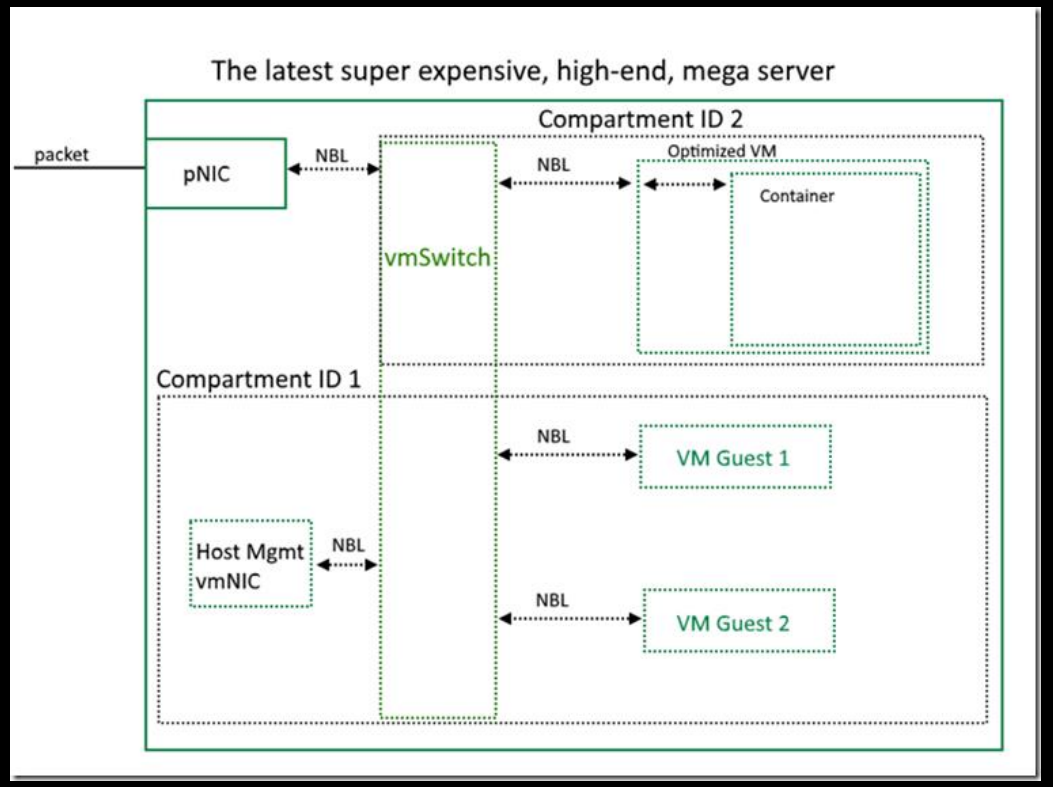
- 읽어보시면 도움되는 글
  - https://bit.ly/2JX3oKY
  - 특히 Part 2, Part 5, Part 6이 HNS와 K8S Networking의 이해의 기본이 됩니다.
- Compartment
  - Windows 내부의 VLAN과 유사한 개념
  - Kubernetes에서는 Service에 대응
- NBL: Net Buffer List
  - WNV 환경 하에서의 패킷으로 이해하면 편리함

Host Network Service (Cont.)

Process Isolation

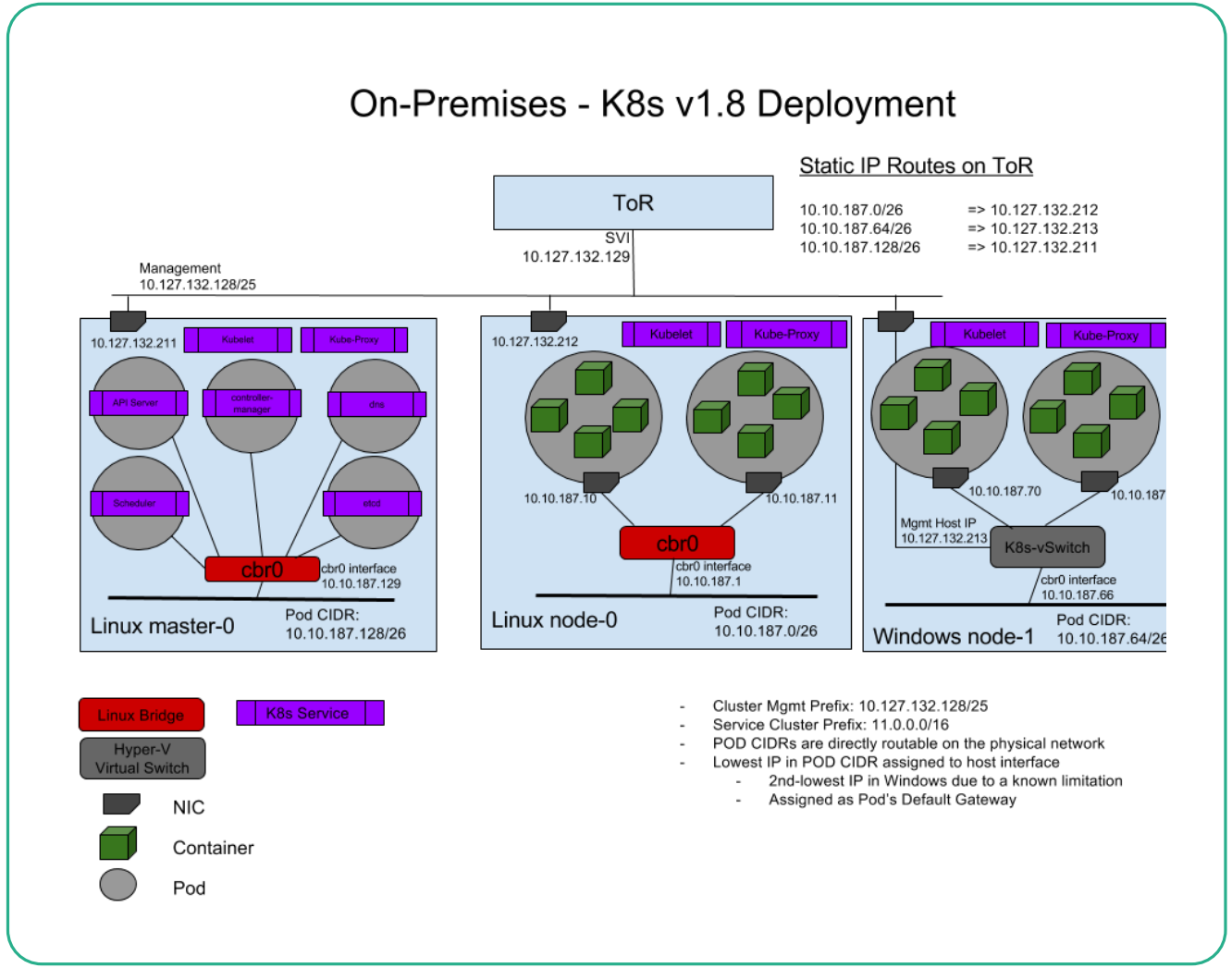


Hyper-V Isolation



네트워킹 구성

- Windows Kubernetes의 네트워크 모델을 크게 두 가지입니다.
  - Transparent NAT
  - WinCNI + L2Bridge
- Transparent
  - 가상 NAT 생성
  - 별도의 Forwarder Adapter 생성 후 Route 테이블 관리
  - 컨테이너 내부의 Outbound 연결에 제약이 있음
- L2Bridge
  - Transparent와 같음
  - WinCNI에서 기본으로 사용하는 네트워킹 방식



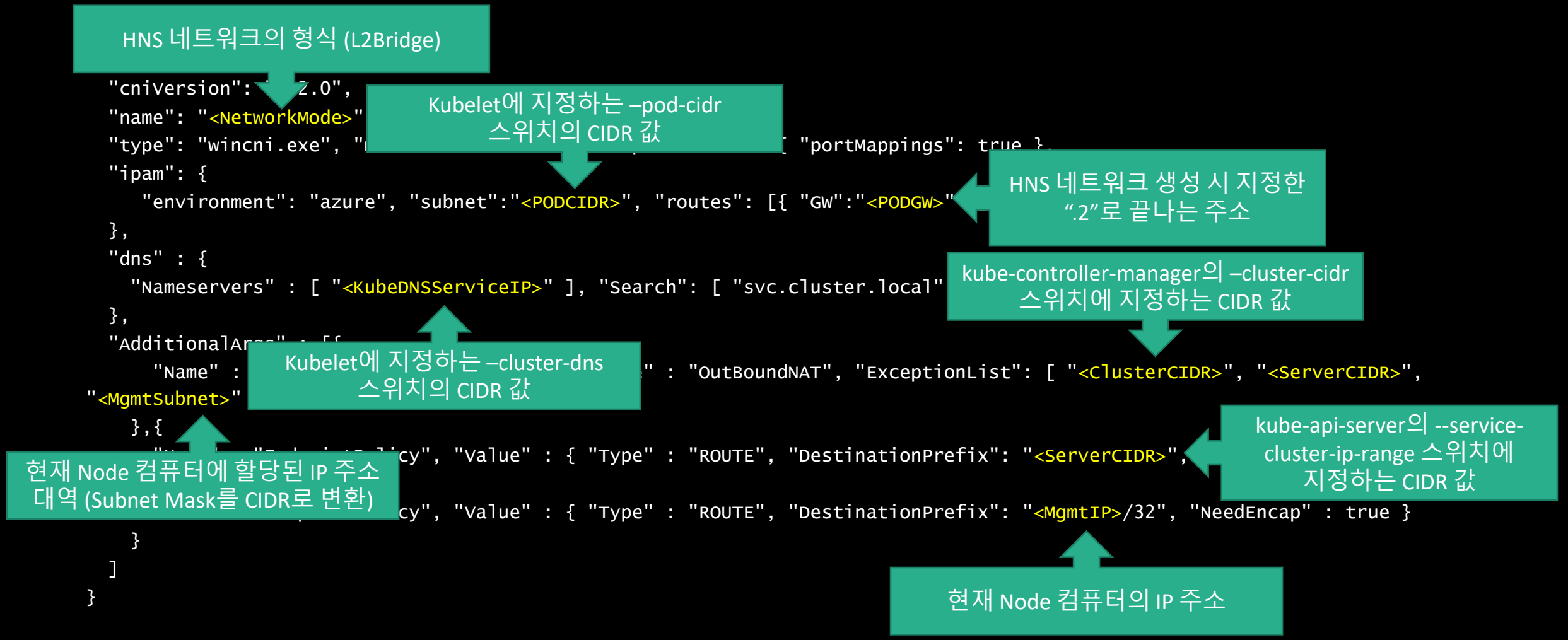
네트워킹 구성 (Cont.)

- Microsoft SDN Git Repo의 Kubernetes 코드 샘플에 HNS 네트워크 제어 모듈이 들어있음
- HNS 네트워크 제어 모듈을 이용하여 새 L2Bridge HNS 네트워크를 생성
- L2Bridge HNS 네트워크에 현재 Worker Node를 연결하기 위하여 새 HNS Endpoint를 생성하고 만든 네트워크에 Attach
- CNI 플러그인 설정 파일 수정
- Kubelet, Kubeproxy 시작
- 수동 라우팅 정보 추가

네트워크 구성 (Cont.)

```
{
  "cniVersion": "0.2.0",
  "name": "<NetworkMode>",
  "type": "wincni.exe", "master": "Ethernet", "capabilities": { "portMappings": true },
  "ipam": {
     "environment": "azure", "subnet":"<PODCIDR>", "routes": [{ "GW":"<PODGW>" }]
  },
  "dns" : {
    "Nameservers" : [ "<KubeDNSServiceIP>" ], "Search": [ "svc.cluster.local" ]
  },
  "AdditionalArgs" : [{
      "Name" : "EndpointPolicy", "Value" : { "Type" : "OutBoundNAT", "ExceptionList": [ "<ClusterCIDR>", "<ServerCIDR>",
"<MgmtSubnet>" ] }
    },{
      "Name" : "EndpointPolicy", "Value" : { "Type" : "ROUTE", "DestinationPrefix": "<ServerCIDR>", "NeedEncap" : true }
    },{
      "Name" : "EndpointPolicy", "Value" : { "Type" : "ROUTE", "DestinationPrefix": "<MgmtIP>/32", "NeedEncap" : true }
    }
  ]
}
```

네트워크 구성 (Cont.)


HNS 네트워크의 형식 (L2Bridge)
Kubelet에 지정하는 –pod-cidr 스위치의 CIDR 값
HNS 네트워크 생성 시 지정한 ".2"로 끝나는 주소
kube-controller-manager의 –cluster-cidr 스위치에 지정하는 CIDR 값
Kubelet에 지정하는 –cluster-dns 스위치의 CIDR 값
현재 Node 컴퓨터에 할당된 IP 주소 대역 (Subnet Mask를 CIDR로 변환)
kube-api-server의 --service-cluster-ip-range 스위치에 지정하는 CIDR 값
현재 Node 컴퓨터의 IP 주소
"cniVersion": "0.2.0",
"name": "<NetworkMode>"
"type": "wincni.exe",
"portMappings": true }
"ipam": {
"environment": "azure", "subnet":"<PODCIDR>", "routes": [{ "GW":"<PODGW>"
},
"dns" : {
"Nameservers" : ["<KubeDNSServiceIP>"], "Search": ["svc.cluster.local"
},
"Name" :
" : "OutBoundNAT", "ExceptionList": ["<ClusterCIDR>", "<ServerCIDR>",
"<MgmtSubnet>"
},{
cy", "Value" : { "Type" : "ROUTE", "DestinationPrefix": "<ServerCIDR>",
cy", "Value" : { "Type" : "ROUTE", "DestinationPrefix": "<MgmtIP>/32", "NeedEncap" : true }
}
]
}

Demonstration

Windows K8S 노드 구성 살펴보기

Demonstration

실제 IIS Pod 실행해보기

Windows 컨테이너 사용 시 주의 사항

- Control Plane 서비스를 제공할 Master Node는 반드시 Linux
- 그 외에는 상황에 따라 Linux Only, Windows Only, Hybrid 구축
- 컨테이너 버전과 호스트 OS 버전과의 상관 관계
- Windows Node의 OS는 1803 권장
  - Azure, AWS 모두 사용 가능
- LTSB는 Windows Server 2019를 기다려 주세요.
  - 2016도 사용은 가능하나,
  - 제약이 많고 성능이 떨어집니다.

Linux Container on Windows (LCOW)

- 아직 개발 중인 기술
- Hyper-V Isolation을 응용하여 부팅 대상을 Windows 커널이 아닌 Linux로 함
- 중첩 가상화를 쓸 수 있는 환경이 필요
  - Azure: 3세대 VM 이상
  - AWS: i3.baremetal 혹은 유사 인스턴스
  - 기타 베어메탈 제공 클라우드 전체
- 이 기술이 제공되면 Windows Node에 대한 Taint를 뗄 수 있음

한국 Azure 사용자 그룹

- 국내에서 가장 많은 Azure 관련 이벤트, 교육을 진행하는 커뮤니티입니다.
- 매달 Saturday Azure Live (이하 Azure Live) Hands-on Lab 행사를 진행합니다.
  - Hands-on-Lab 실습을 위한 계정을 무료로 제공합니다.
- https://www.fb.com/groups/krazure

마무리

Windows Server 2019가 출시될 때까지는 아직 많이 개선될 여지가 남아있습니다.

Windows 기반의 애플리케이션 개발자들에게 새로운 기회가 될 수 있습니다.

더 살펴보실 키워드: Nano Server, OneCore API, .NET Core, Docker for Windows, Windows Container

고맙습니다!