02. 클라우드 인프라

레거시 시스템 현대화를 위한 클라우드 기반의 기술 전환 방법론을 정의합니다.
마이크로 서비스를 통한 레거시 시스템의 변화 적용 , 마이크로 서비스를 운영하기 위한 플랫폼으로서의 Kubernetes 시스템 그리고 Kubernetes Node를 가상화하고 다이나믹하게 제공하기 위한 가상화 인프라 기술이 필요하다

01.OK ! Stangler Pattern 변화를 위한 기술 조건은 ?

MSA를 해야 하는 이유를 살펴 보았다 http://web.joang.com:8083/books/01-why

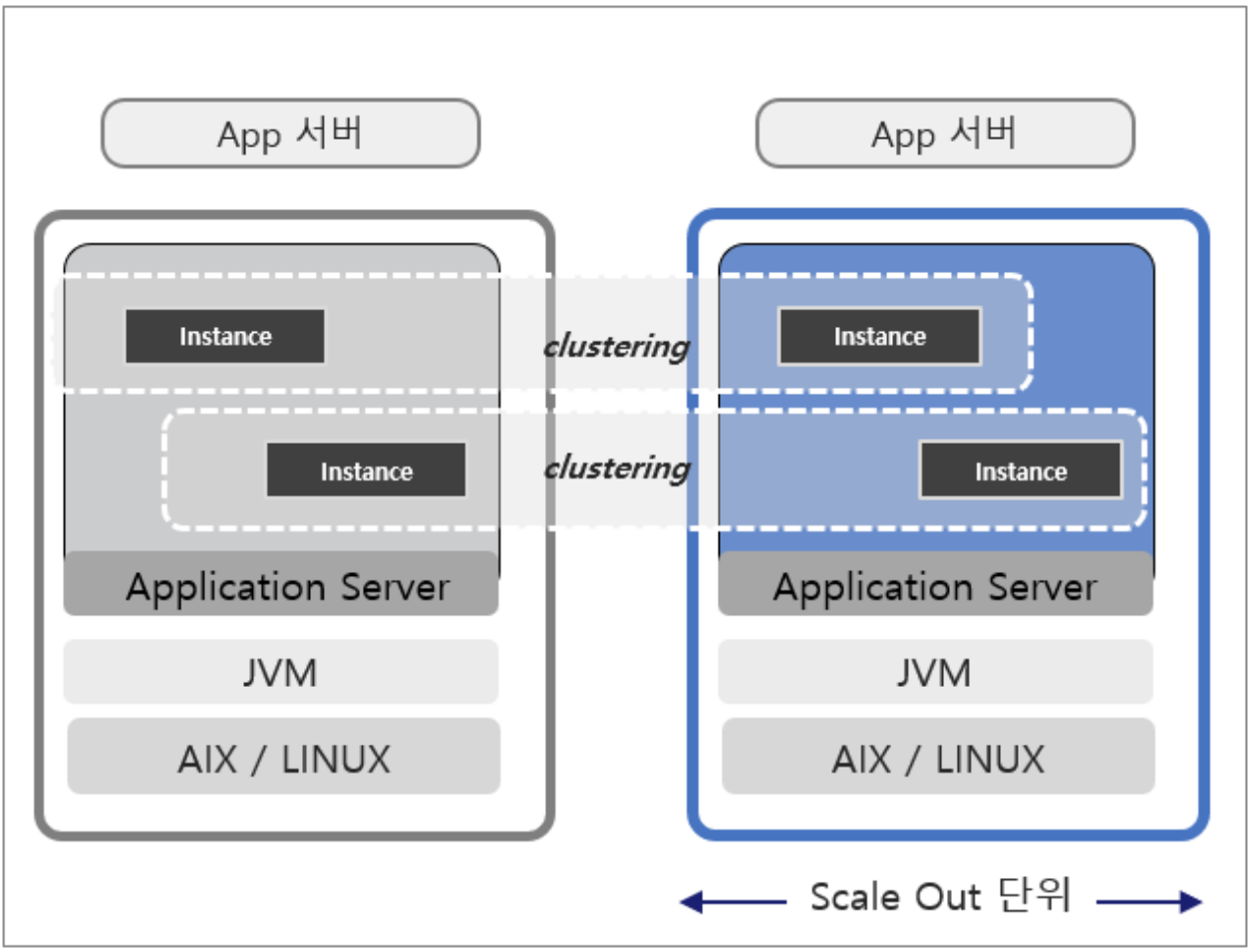
그러면 MSA를 하기위한 조건은 어떻게 되는가 ?

첫째로 우리는 로 우리의 시스템을 재편해야 하고, 그러한 마이크로한 서비스는 (Pod)될 것이다. 그리고 그러한 하나하나의 서비스에 일정한 분량의 해야 하고 그리고 무수히 만들어지는 서비스를을 유지적으로 관리할 수 있는 환경을 구축해야 한다.

1. 그럼 모노로틱한 서비스를 마이크로한 서비스로 분리할 때 마이크로 서비스는 어떻게 나누어질 수 있을까 ?

모노로틱한 개발은 명시적이고 가시적이고 온전한 개발에 용이하다 . 우리가 모노로틱한 개발에서 마이크로서비스로 전환하고자하는 것은 모노로틱 애플리케이션의 한계 때문이다.
어떻게 나눌것인가 ? 결론부터 말하자면 나누지 말자 이다. 너무도 명시적인 분리가 사실 모노로틱에 존재한다.

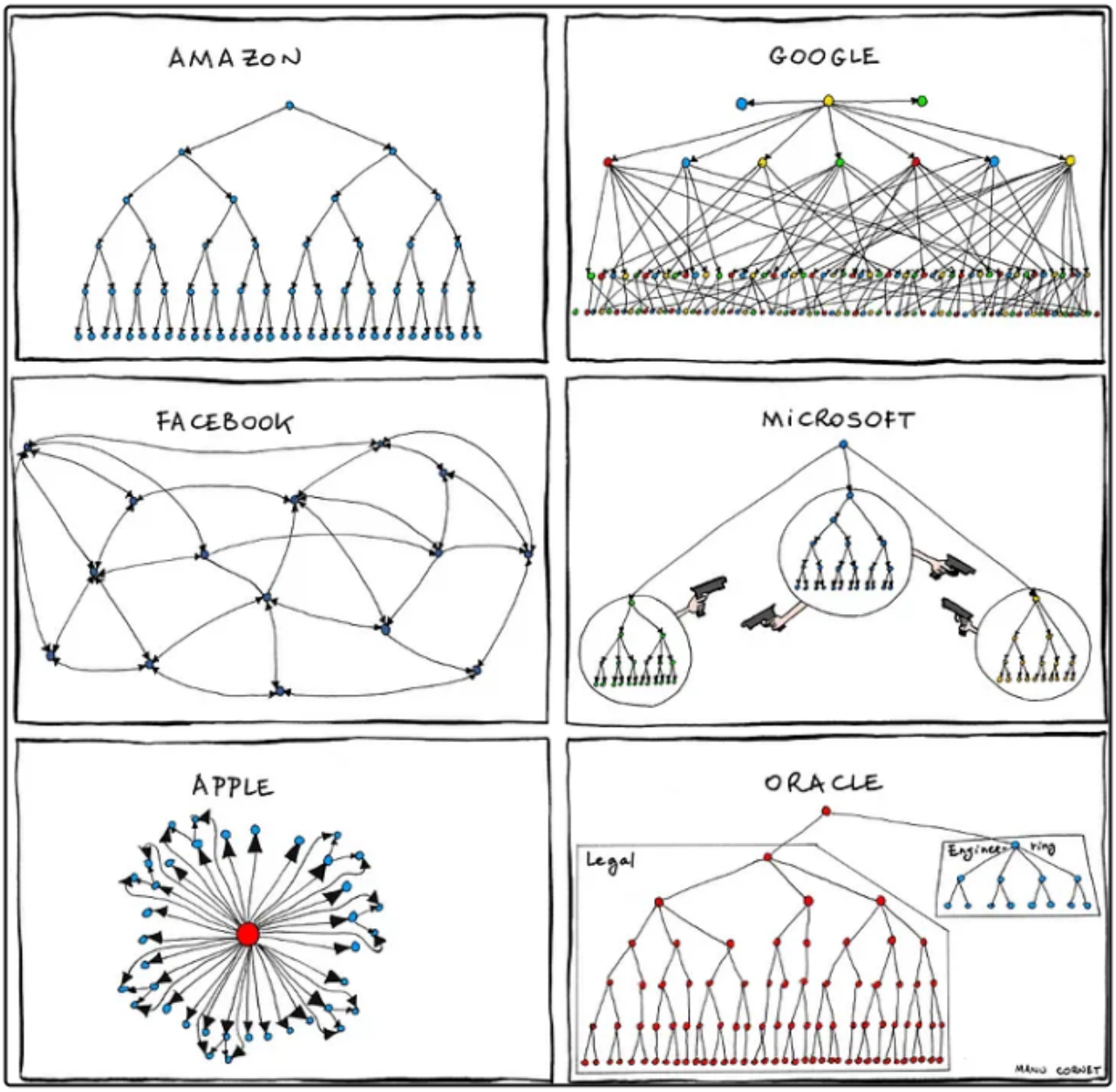
On-Premise 또는 IaaS 기반의 어플리케이션 아키텍처



우선은 오랜 세월과 개발 관계에서 자연스럽게 만들어진 코웨이의 법칙을 인정하자 그리고 그것이 우리 조직과 운영에 최적이라는 사실을 용감하게 인정하고 시작하자

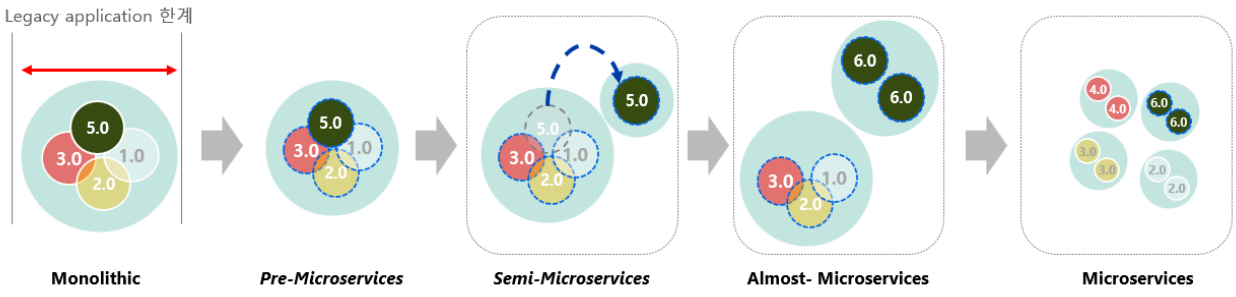
“ 시스템을 설계하는 조직은 어떤 조직이든
그 조직의 소통 구조를 닮은 구조를 가진 시스템으로
설계할 것이다. ”

멜빈 콘웨이, 1967



마이크로소프트 ㅋㅋ

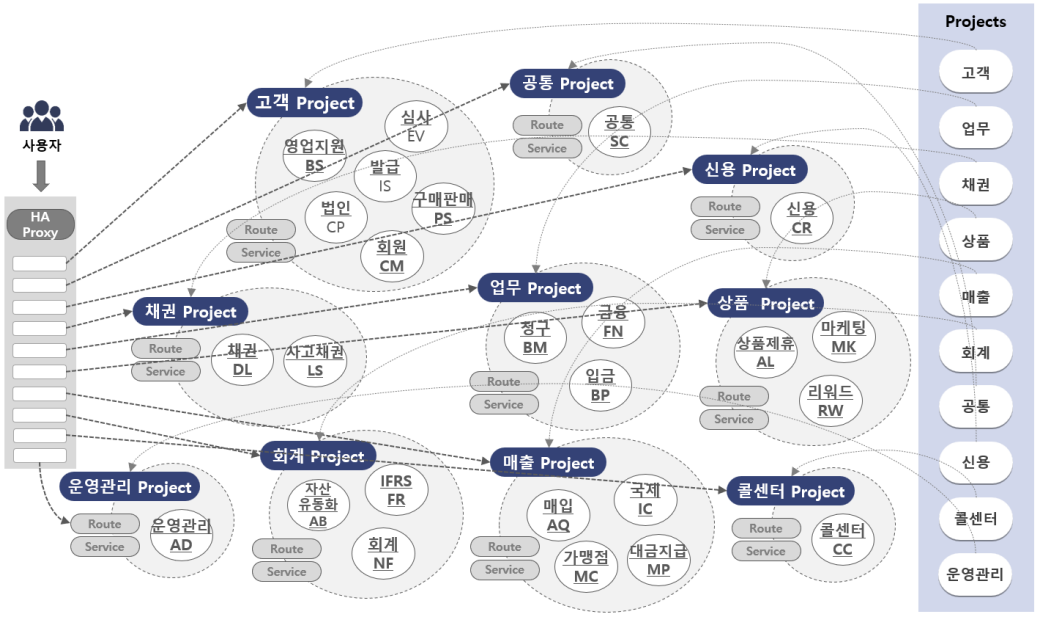
그리고 그 과정을 통하여 마이크로서비스를 구축한다 . 우선은 운영성을 고려하고 안전한 마이크로서비스로의 전환을 만든다.
이제 다소 거북한 마이크로서비스에 대한 부담이 느껴진다면 과감하게 시작하면 될것이다.



마이크로서비스 저니 [Journey to Microservices.pptx]

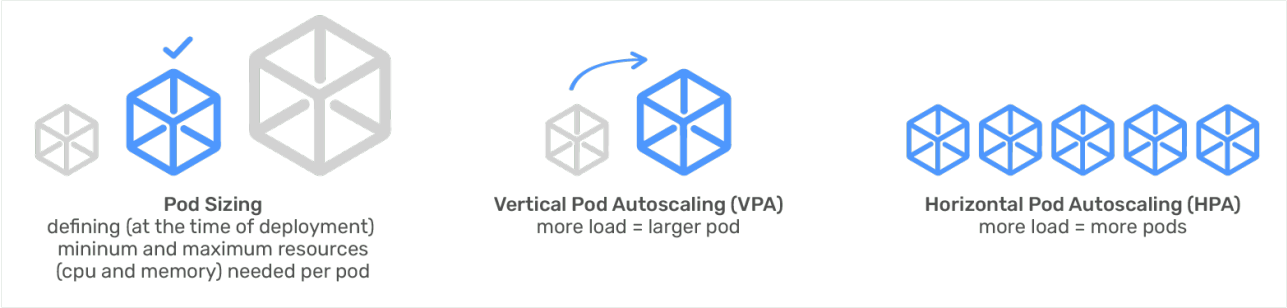
2. 마이크로 서비스들이 하나 둘이면 크게 문제가 될 여지는 없지만, 마이크로서비스가 100여개가 넘어갈 때 어떻게 해야 하는가 ?

상기한 바와 같이 이제 콘테이너라고 하는 놈들이 많아질 것이다. 더불이 콘테이너 뿐만 아니라 콘테이너에 외부 호출을 연결하여 주는 네트워크까지 복잡하게 고려를 해야 한다.
 한때 XX면제점에서 Docker 기반으로 개발을 하였을 때, 내가 비관적으로 보았던 이유 중의 하나가 콘테이너를 관리하는 것이 없이 저렇게 많은 node들에 docker run을 하는 것이 맞는가 하는 것이었다.
 이후 내가 XX카드사의 채널과 기간계를 구축하면서 콘테이너 플랫폼인 OCP의 덕을 톡톡히 보기는 하였다. 이후로 PaaS 구축의 전도사가 되었다.



 각각의 Pod들은 서로 다른 리소스를 사용하고 각각의 라이프 사이클과 더불어 기획 개발 테스트 운영의 주체가 별도로 구성되어 진다.

네오(키아누리브스)가 잘싸웠지만 스미스 요원은 우리를 경악하게 하였다 네오는 PaaS 였으면 굳이 힘들지 않았을 텐데 싶다. 동양은 손오공의 머리키락일텐데, 사실 황당하지만 오래전부터 사실 누구나 그랬으면 싶은게 자기복제 기술이 아닐까 한다.



3. 이제 마이크로한 서비스는 비교적 모노로틱한 서비스에 비하여 작게 나누어 지는데 그럼 작은 Linux 서버나 UNIX 서버를 그때 그때 사야 하는가 ?

이제 모든게 준비되었다면 그 작은 콘테이너가 독자적으로 동작할 수 있는 환경의 자원이 필요하다.
그때 그때 작은 자원의 활당과 회수 그리고 확장에 용이한 구조의 리소스 자원 관리는 무엇인가 ?

당연히 누구나 예측할 수 있는 클라우드 환경이다. 클라우드 환경은 아래와 같이 다양한 클라우드가 존재하고 , 기업에서는 가장 적합한 환경의 클라우드를 구축하면 된다.

[클라우드 분류]		Deploy Model (발전 방향 →)		
		IaaS (Infrastructure as a Service)	**PaaS** (Platform as a Service)	**SaaS** (Software as a Service)
Service Model (외부 ↔ 내부)	**Public** (Off-Premise)	• 인터넷 상에서 논리적인 자원(virtual Machine)을 제공 • 대부분의 인터넷 클라우드 업체에서 일반적으로 제공	• **아마존 EKS(Elastic Kubernetes Service) , Google GKE(Google Kubernetes Engine) 등** • 인터넷 상에서 서비스 Platform을 제공	• 클라우드 기반 어플리케이션 서비스 제공 • 세일즈포스 CRM, 아마존 AI서비스, Google 어시스턴스, 이미지 인식, 음성인식 서비스 등
	Hybrid	• 인터넷 상의 가상화 인프라서비스와 기업 내부의 가상화 인프라가 병존 하는 형태의 구성	• 내부 시스템과 인터넷 클라우드 제공 환경을 복합 구성 • 예) 내부 시스템 + BigData , 백오피스 + 프론트 오피스, 해외 사이트 등	• 클라우드 서비스를 내부 시스템이 사용하거나 내부 시스템의 서비스를 퍼블릭에 제공 • 예) OO맴버스의 BigData 서비스 등
	Private (On-Premise)	• 기업 내부의 물리 자원에 대한 가상화를 통하여 논리적 서버를 제공 • 기본은 기업이 인프라 자원을 도입하고 가상화 하여 제공 • **VMWare , OpenStack, Nutanix 등**	• 기업 내부 시스템을 기반으로 개발/운영 플랫폼을 제공하는 형태 • **Kubernetes, OpenShift, Cloud Foundry 등**	• 대규모 기업 시스템의 경우는 공통 서비스 영역을 내부 클라우드에 제공하는 형태 • SSO, 공통BIZ, 암복호화 서비스, 결재 서비스 등 기업

HCI (Hyper Converged Infra) 및 CI (Converged Infra) IT 인프라 모두 스토리지, 컴퓨팅 및 네트워킹을 통합합니다.
하이퍼 컨버지드 시스템은 소프트웨어를 활용한 통합으로 하드웨어에 구애받지 않지만, 컨버지드 솔루션은 하드웨어에 종속됩니다.

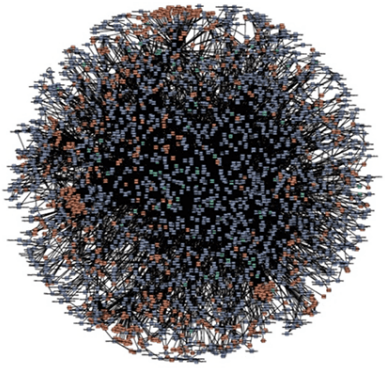
이제 준비가 되었으면 클라우드 기반의 가상 인프라자원을 필요한 만큼 신속하게 요청한다.

4. 주의할 점

마이크로서비스는 역설적이게도 마이크로서비스로 만들지 말라는 것이다 . 즉 모노로틱한 서비스를 나누어 효율적으로 진화해 나가는 것이 마이크로서비스이기 때문에 미리 진화한 모습으로 만들지 말자 어떻게 진화할 지 모르는데 ... 단 아주 중요한 것은 모노로틱하게 우선 만들더라도 인터페이스를 활용하여 호출하는 형태로 나들라는 것이다 그래야 마이크로로 진화하는데 용이하다.

둘째 마이크로서비스는 결코 마이크로만하지 않다는 것이다. 고정관념을 가지고 작은 것으로 나누는 것이 효율적이라는 선부른 생각은 버려야한다. 결코 마이크로서비스가 작은 단위의 서비스가 아니라는 것이다.

Netflix가 되지 말아야 합니다. 우리 회사가 Netflix 는 아니니까



amazon.com

NETFLIX

02. 업무를 모듈화하는, 서비스의 정체성 얘기 또 해야 하나 ? 이전과 뭐가 다른가 ?

마이크로 서비스 아키텍처

http://web.joang.com:8083/books/msa-showcase

마이크로서비스로 구축하기 위해서 기업 업무 설계를 어떻게 해야 하는 것일까 --> 이거 너무 오래되고 자주하는 얘기인데 답없는 얘기던데 마이크로서비스 세상에서는 어떻게 논의되고 있고, 어떻게 실행하는 것이 현실적일까 생각 해 봅니다.
마이크로 서비스를 위한 DDD 방법론의 사전적인 의미

우리글 : http://web.joang.com:8083/books/ddd/page/ddd

이전에 설명한 이유로의 마이크로서비스 구현의 설계라는 관점에서 아래 사항이 이전의 다른 방법론과 달라야 한다.

- 모델은 정의하는 것이 아니라 지속적으로 변화하는 것이다. **--> 모델이 여러 관점에서 수정, 보완되어야 한다. 마치 마인드맵과 같이**
- 업무 전문가와 개발 전문가가 같이 설계해야 한다. 같은 팀에 소속된 두 전문가가 동일한 비즈니스 언어를 구사해야 한다. **--> 지금까지는 모두 기술 언어이었다 그래서 아무리 정의해도 업무 전문가는 이해하지 못하고 자신들의 언어인 엑셀과 PPT로 업무를 정의한다. 특히나 문제는 이러한 방법은 이해는 쉽지만 기술적으로 많은 헛점들을 명확하게 하지 않는다는 것이다. 그래서 Ubiquitous Language라고 하는데 이거 말장난 같아서 별로 , 사실 엔지니어만 이해하는 언어임 (문제가 많음) 그래도 많이 발전해서 업무 담장도 이해 할 듯**
- **이제 도메인은 분리되어서 lifecycle을 갖게 되고, 업무적으로도 이제 분리 독립적으로 정의**되어야 하지만 전체적으로는 일관성이 있는 구조여야 한다.
  이전처럼 전지에 모두 모여서 도시계획과 같은 깨알같은 그림은 이제 한계가 있다. **다 자기 것 알아서 그리고 인터페이스만 이야기 하자**
- 기존에 마이크로서비스로 설계하는 시작점이 아니라 기존의 모노로틱한 Application 을 분해하는 측면에서는 **break down(무너트리듯이 세분화)하는 과정이 자연스러워야 한다**. **--> 즉 한덩어리큼직한 놈과 작은 알갱이들의 interaction이 자연스럽게 표현되고 이해되어야 한다.**

이제 방법론에 대하여 얘기해 보자 , 방법론을 이야기 해 보자 -- 내가 가장 싫어하는방법론 얘기 , 학교가는 얘기, 하지만 어떤식으로든지 업무를 정의하고 업무 담당자와 프론트 개발자와 백엔드 개발자와 AA, SA 도두가 이해를 해야 하니까

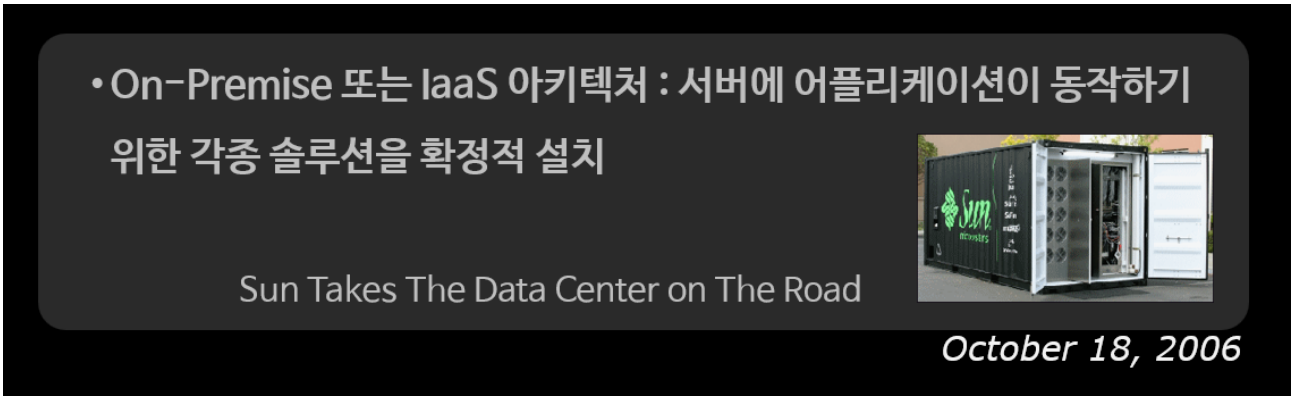
이거는 다른 사람한테 맡긴다.

03. 콘테이너? 뭔가 캡슐화하여 하나의 독립적으로 움직이는 것? 스미스 요원 만들기

콘테이너란 무엇인가 ?

근데 왜 콘테이너라는 얘기가 나오는가 ? 그래 서비스 업무를 나눈 것들을 독립적으로 패키징하여 독립적으로 동작할 수 있도록 하는 것이 중요하겠네

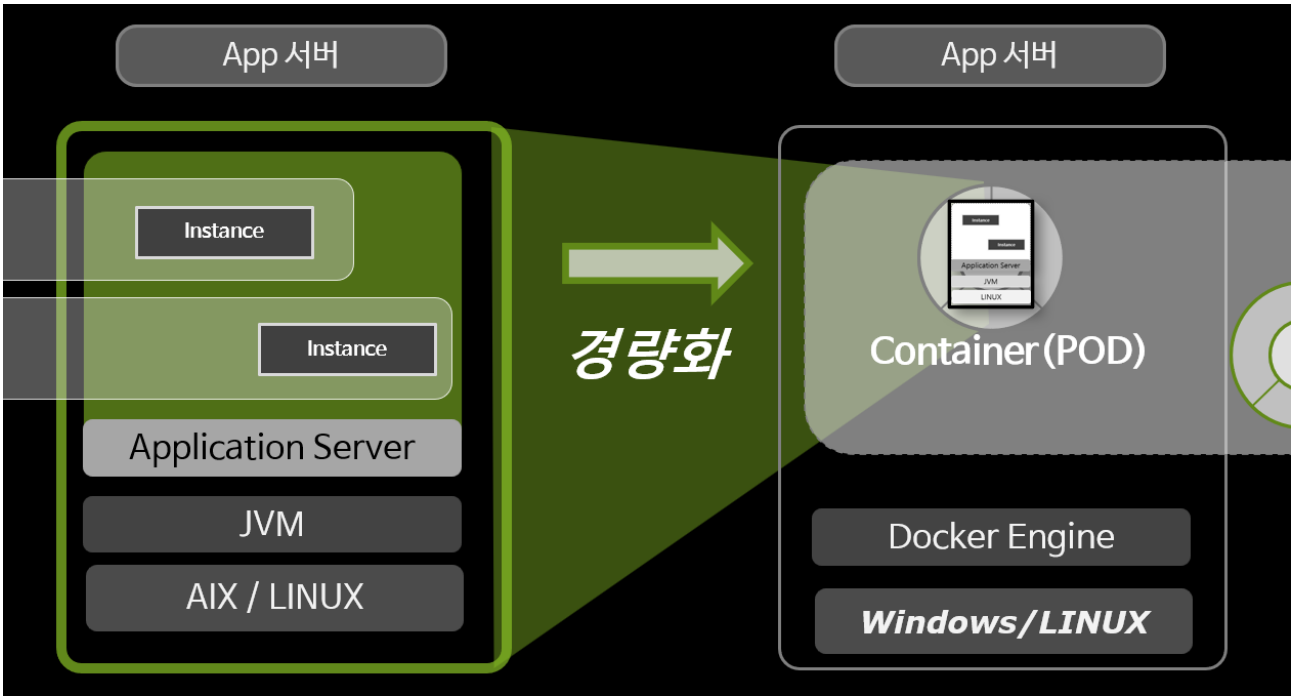
portability 이 개념이 중요하겠습니다.



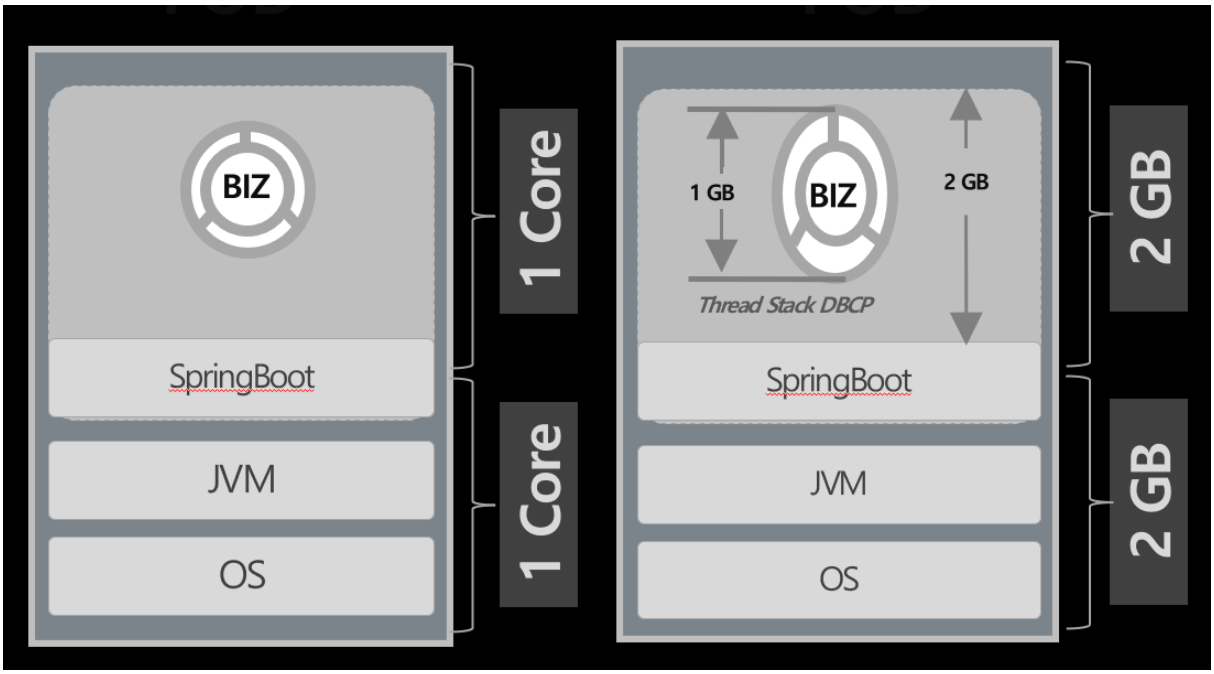
이제 콘테이너화 한다는 것은 어느 곳에도 동일한 조건의 기본적인 환경만 주어진다면 하나 이상이 동작할 수 있는 환경을 말합니다.
java를 생각해보면 자바의 **WORA**(Write Once, Run Anywhere)와 비슷한 개념이다. 제임스고슬링이 전자 장비 회사(썬)를 다닐 때 다양한 장비에 계속해도 비슷한 기능을 개발하는 것이 불합리하여 JVM만 각 장비별로 구축하고 원 소스는 하나로 유지하고자 하여 만들었다고 하는데 유사한 이유가 될 것이다.

잘 패키징된 콘테이너는 이제 표준을 준수하는 어떤 곳에서도 하나 이상이 동작하기 때문에 복제, 확장, 중복, 축소 등 매우 탄력적인 서비스의 기능을 수행한다.

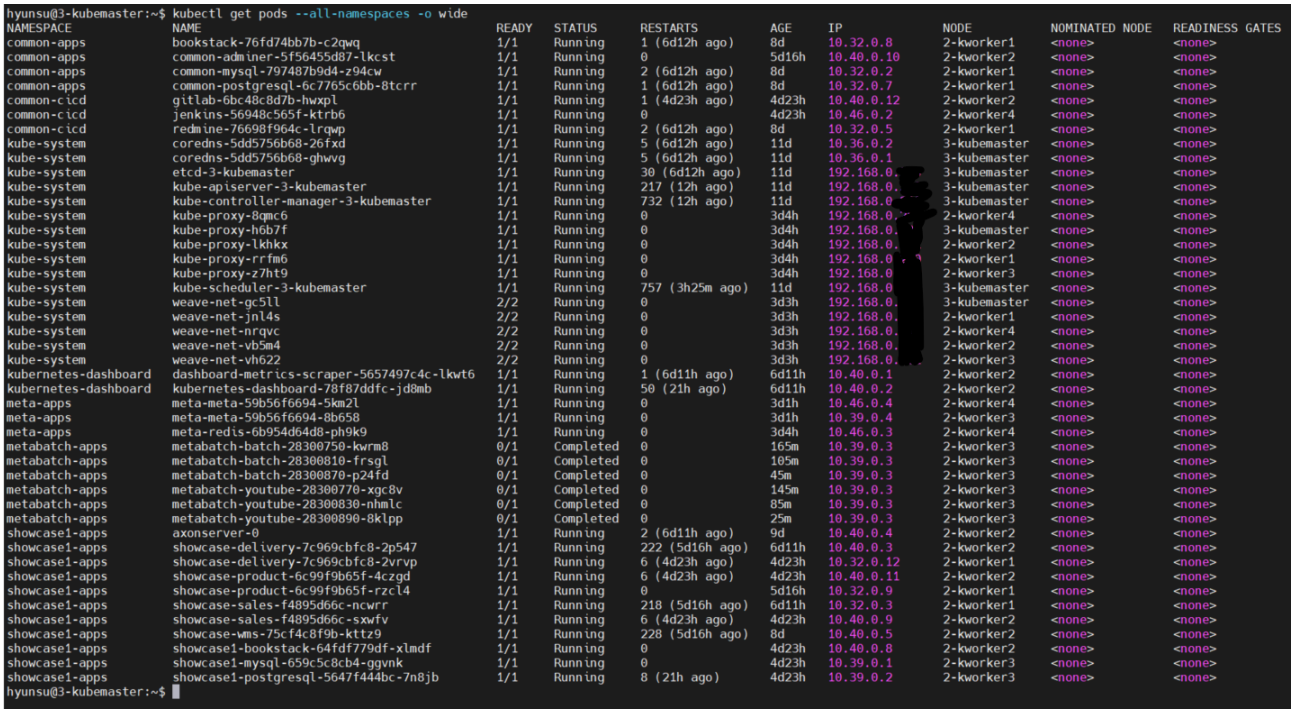
그럼 콘테이너의 구조는 어떠한가 ?



과거의 OS를 포함한 was 환경이 작은 파일 하나로 줄어들어 패키징된다고 생각하면 된다.



콘테이너의 코어 수와 메모리 예시

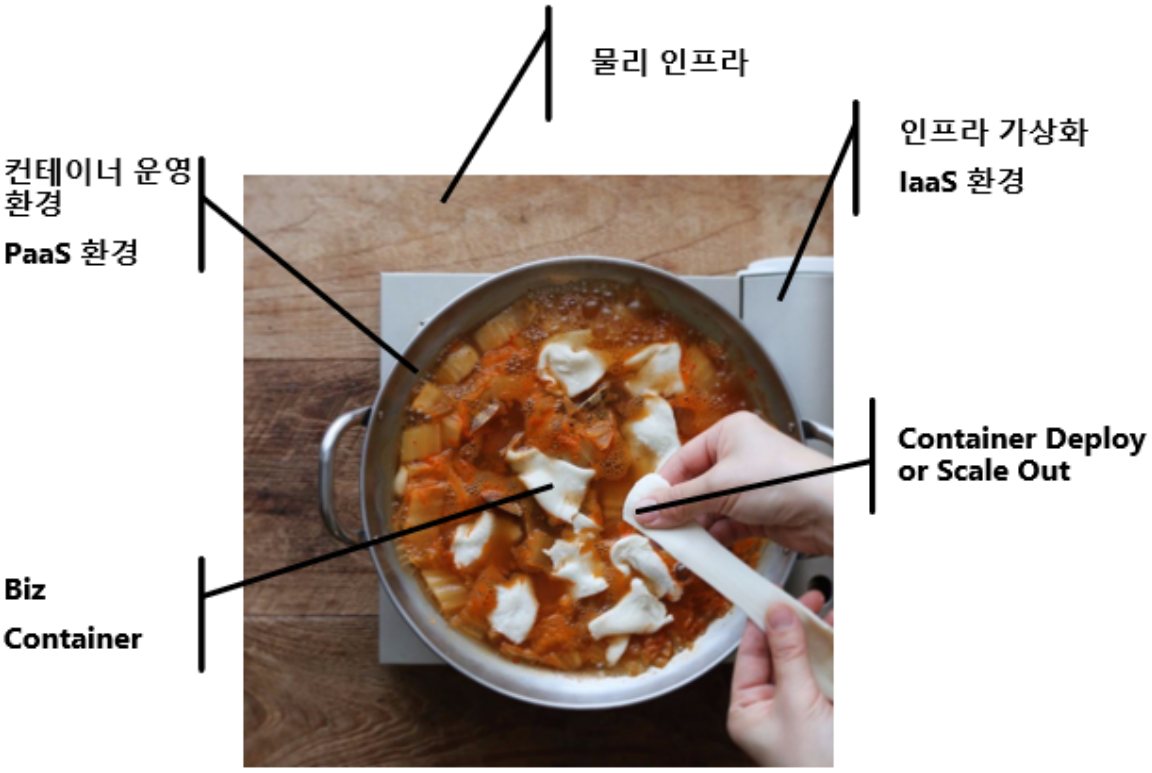


```
hyunsu@3-kubemaster:~$ kubectl get pods --all-namespaces -o wide
NAMESPACE              NAME                                        READY   STATUS      RESTARTS          AGE     IP              NODE          NOMINATED NODE   READINESS GATES
common-apps            bookstack-76fd74bb7b-c2qwq                  1/1     Running     1 (6d12h ago)     8d      10.32.0.8       2-kworker1    <none>           <none>
common-apps            common-adminer-5f56455d87-lkcst             1/1     Running     0                 5d16h   10.40.0.10      2-kworker2    <none>           <none>
common-apps            common-mysql-797487b9d4-z94cw               1/1     Running     2 (6d12h ago)     8d      10.32.0.2       2-kworker1    <none>           <none>
common-apps            common-postgresql-6c7765c6bb-8tcrr          1/1     Running     1 (6d12h ago)     8d      10.32.0.7       2-kworker1    <none>           <none>
common-cicd            gitlab-6bc48c8d7b-hwxpl                     1/1     Running     1 (4d23h ago)     4d23h   10.40.0.12      2-kworker2    <none>           <none>
common-cicd            jenkins-56948c565f-ktrb6                    1/1     Running     0                 4d23h   10.46.0.2       2-kworker4    <none>           <none>
common-cicd            redmine-76698f964c-lrqwp                    1/1     Running     2 (6d12h ago)     8d      10.32.0.5       2-kworker1    <none>           <none>
kube-system            coredns-5dd5756b68-26fxd                    1/1     Running     5 (6d12h ago)     11d     10.36.0.2       3-kubemaster  <none>           <none>
kube-system            coredns-5dd5756b68-ghwvg                    1/1     Running     5 (6d12h ago)     11d     10.36.0.1       3-kubemaster  <none>           <none>
kube-system            etcd-3-kubemaster                           1/1     Running     30 (6d12h ago)    11d     192.168.0.      3-kubemaster  <none>           <none>
kube-system            kube-apiserver-3-kubemaster                 1/1     Running     217 (12h ago)     11d     192.168.0.      3-kubemaster  <none>           <none>
kube-system            kube-controller-manager-3-kubemaster        1/1     Running     732 (12h ago)     11d     192.168.0       3-kubemaster  <none>           <none>
kube-system            kube-proxy-8qmc6                            1/1     Running     0                 3d4h    192.168.0.      2-kworker4    <none>           <none>
kube-system            kube-proxy-h6b7f                            1/1     Running     0                 3d4h    192.168.0.      3-kubemaster  <none>           <none>
kube-system            kube-proxy-lkhkx                            1/1     Running     0                 3d4h    192.168.0.      2-kworker2    <none>           <none>
kube-system            kube-proxy-rrfm6                            1/1     Running     0                 3d4h    192.168.0       2-kworker1    <none>           <none>
kube-system            kube-proxy-z7ht9                            1/1     Running     0                 3d4h    192.168.0       2-kworker3    <none>           <none>
kube-system            kube-scheduler-3-kubemaster                 1/1     Running     757 (3h25m ago)   11d     192.168.0       3-kubemaster  <none>           <none>
kube-system            weave-net-gc5ll                             2/2     Running     0                 3d3h    192.168.0.      3-kubemaster  <none>           <none>
kube-system            weave-net-jnl4s                             2/2     Running     0                 3d3h    192.168.0.      2-kworker1    <none>           <none>
kube-system            weave-net-nrqvc                             2/2     Running     0                 3d3h    192.168.0.      2-kworker4    <none>           <none>
kube-system            weave-net-vb5m4                             2/2     Running     0                 3d3h    192.168.0.      2-kworker2    <none>           <none>
kube-system            weave-net-vh622                             2/2     Running     0                 3d3h    192.168.0.      2-kworker3    <none>           <none>
kubernetes-dashboard   dashboard-metrics-scraper-5657497c4c-lkwt6  1/1     Running     1 (6d11h ago)     6d11h   10.40.0.1       2-kworker2    <none>           <none>
kubernetes-dashboard   kubernetes-dashboard-78f87ddfc-jd8mb        1/1     Running     50 (21h ago)      6d11h   10.40.0.2       2-kworker2    <none>           <none>
meta-apps              meta-meta-59b56f6694-5km2l                  1/1     Running     0                 3d1h    10.46.0.4       2-kworker4    <none>           <none>
meta-apps              meta-meta-59b56f6694-8b658                  1/1     Running     0                 3d1h    10.39.0.4       2-kworker3    <none>           <none>
meta-apps              meta-redis-6b954d64d8-ph9k9                 1/1     Running     0                 3d4h    10.46.0.3       2-kworker4    <none>           <none>
metabatch-apps         metabatch-batch-28300750-kwrm8              0/1     Completed   0                 165m    10.39.0.3       2-kworker3    <none>           <none>
metabatch-apps         metabatch-batch-28300810-frsgl              0/1     Completed   0                 105m    10.39.0.3       2-kworker3    <none>           <none>
metabatch-apps         metabatch-batch-28300870-p24fd              0/1     Completed   0                 45m     10.39.0.3       2-kworker3    <none>           <none>
metabatch-apps         metabatch-youtube-28300770-xgc8v            0/1     Completed   0                 145m    10.39.0.3       2-kworker3    <none>           <none>
metabatch-apps         metabatch-youtube-28300830-nhmlc            0/1     Completed   0                 85m     10.39.0.3       2-kworker3    <none>           <none>
metabatch-apps         metabatch-youtube-28300890-8klpp            0/1     Completed   0                 25m     10.39.0.3       2-kworker3    <none>           <none>
showcase1-apps         axonserver-0                                1/1     Running     2 (6d11h ago)     9d      10.40.0.4       2-kworker2    <none>           <none>
showcase1-apps         showcase-delivery-7c969cbfc8-2p547          1/1     Running     222 (5d16h ago)   6d11h   10.40.0.3       2-kworker2    <none>           <none>
showcase1-apps         showcase-delivery-7c969cbfc8-2vrvp          1/1     Running     6 (4d23h ago)     4d23h   10.32.0.12      2-kworker1    <none>           <none>
showcase1-apps         showcase-product-6c99f9b65f-4czgd           1/1     Running     6 (4d23h ago)     4d23h   10.40.0.11      2-kworker2    <none>           <none>
showcase1-apps         showcase-product-6c99f9b65f-rzcl4           1/1     Running     0                 5d16h   10.32.0.9       2-kworker1    <none>           <none>
showcase1-apps         showcase-sales-f4895d66c-ncwrr              1/1     Running     218 (5d16h ago)   6d11h   10.32.0.3       2-kworker1    <none>           <none>
showcase1-apps         showcase-sales-f4895d66c-sxwfv              1/1     Running     6 (4d23h ago)     4d23h   10.40.0.9       2-kworker2    <none>           <none>
showcase1-apps         showcase-wms-75cf4c8f9b-kttz9               1/1     Running     228 (5d16h ago)   8d      10.40.0.5       2-kworker2    <none>           <none>
showcase1-apps         showcase1-bookstack-64fdf779df-xlmdf        1/1     Running     0                 4d23h   10.40.0.8       2-kworker2    <none>           <none>
showcase1-apps         showcase1-mysql-659c5c8cb4-ggvnk            1/1     Running     0                 4d23h   10.39.0.1       2-kworker3    <none>           <none>
showcase1-apps         showcase1-postgresql-5647f444bc-7n8jb       1/1     Running     8 (21h ago)       4d23h   10.39.0.2       2-kworker3    <none>           <none>
hyunsu@3-kubemaster:~$
```

우리집 컴퓨터에 동작하는 콘테이너들

- 개인 Bookstack : http://web.joang.com:8084/shelves
- Showcase용 Bookstack : http://web.joang.com:8083
- 메타 시스템 : http://web.joang.com:8000/meta/login.do + 배치 2개 (메타 백업, Youtube Download 배치)
- DB 관리툴 (Adminer) http://192.168.0.200:8082
- Postgresql (2EA)
- Mysql (2EA)
- GitLab : http://192.168.0.200:8085/users/sign_in
- Jenkins : http://192.168.0.200:30001
- redmine : http://192.168.0.200:30002
- showcase application 5 services X 2 replicaset : http://192.168.0.200:8024
- Kubernetes Dashboard : https://192.168.0.200:31055/#/login
-

하나의 PC에 위 서비스가 모두 돌고 있다는 ㅎㅎ


물리 인프라
인프라 가상화
IaaS 환경
컨테이너 운영 환경
PaaS 환경
Container Deploy or Scale Out
Biz
Container

04. 인프라 가상화 <- 이거 이제는 default 아닌가 ?

Infrastructure virtualization

가상 인프라(Virtual Infrastructure)는 기업 IT 환경을 구성하는 소프트웨어 정의 구성 요소의 집합

가상 인프라는 소프트웨어를 통해 물리적 자원과 동일한 IT 능력을 제공하며, IT 팀이 기업의 다양한 요구 사항에 따라 빠르게 가상 자원을 할당할 수 있도록 합니다.

물리적 하드웨어와 운영 체제를 분리함으로써 가상 인프라는 조직이 더 큰 IT 자원 활용도, 유연성, 확장 가능성 및 비용 절감을 달성합니다.

인프라의 가상화는 기업 내부의 가상화 또는 클라우드 제공 업체(AWS, Azure등)를 이용할 수 있습니다.

왜 ? Virtual infrastructure

가상 인프라에서는 물리적 하드웨어(서버, 스토리지, 네트워킹 등)가 소프트웨어 레이어(운영 체제 및 응용 프로그램 포함)에서 추상화되어 분리됩니다. 이 분리로 인해 IT 자원을 관리할 때 더 큰 유연성과 민첩성을 얻을 수 있습니다.

가상화 기술을 사용하면 IT 팀이 가상 자원을 신속하게 효율적으로 할당할 수 있습니다. 이는 가상 머신(VM) 또는 컨테이너를 물리적 하드웨어를 물리적으로 추가하거나 제거하지 않고 프로비저닝, 수정 또는 제거할 수 있음을 의미합니다. 이러한 민첩성은 비즈니스 요구 사항의 변화에 대응하는 데 중요합니다.

가상화는 여러 개의 가상 인스턴스가 하나의 물리적 서버에서 실행될 수 있으므로 자원 활용률을 향상시킬 수 있습니다. 이러한 통합으로 하드웨어 자원의 비효율적 사용을 줄여 비용 절감을 이끌어냅니다.

가상 인프라는 수요에 따라 쉽게 확장하거나 축소할 수 있습니다. 더 많은 컴퓨팅 파워, 스토리지 또는 네트워킹 자원이 필요한 경우 추가 가상 인스턴스를 생성하고, 수요가 감소하면 자원을 해제할 수 있습니다.

물리적 하드웨어에 대한 의존도를 줄이고 자원 사용을 최적화함으로써 가상 인프라는 중요한 비용 절감을 가져올 수 있습니다. 특히 소기업이 방대한 물리적 하드웨어 투자를 할 예산이 없는 경우에 유용합니다.

가상화 기술에는 고가용성 및 재해 복구 기능이 포함되어 있습니다. VM은 물리적 호스트 간에 이동할 수 있으며 스냅샷을 촬영하여 백업 및 복구 프로세스를 용이하게 할 수 있습니다.

가상화는 가상 인스턴스 간의 더 나은 격리를 허용합니다. 보안 정책과 액세스 제어를 가상화 레이어에 적용하여 전체적인 보안을 향상시킬 수 있습니다.

가상 인프라는 테스트 및 개발 환경에 매우 유용합니다. 개발자는 프로덕션 환경에 영향을 미치지 않고 고립된 가상 인스턴스를 생성하여 애플리케이션을 개발, 테스트 및 디버깅할 수 있습니다.

가상 인프라는 클라우드 서비스와 통합할 수 있으며 하이브리드 IT 환경을 만들 수 있습니다. 이로써 조직은 클라우드의 확장성과 유연성을 활용하면서 일부 리소스를 온프레미스에서 제어할 수 있습니다.

전반적으로, 가상 인프라는 현대 IT 환경의 기본 구성 요소로, 유연성, 자원 최적화 및 비용 효율성 측면에서 다양한 이점을 제공합니다. 그것은 모든 크기의 비즈니스의 동적이고 진화하는 요구 사항을 충족시키는 중요한 역할을 합니다.