DEVIEW 2020

Container 보안 이야기 (docker, kubernetes)

심재훈 NAVER Cloud

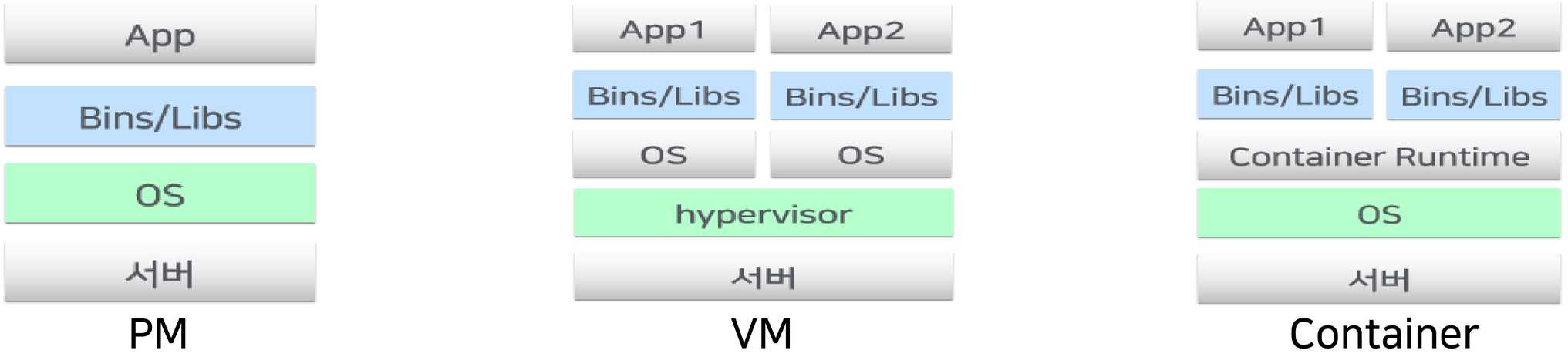
CONTENTS

1. container 환경이란?

1.1 일관성 있으며 폭넓은 구독 환경

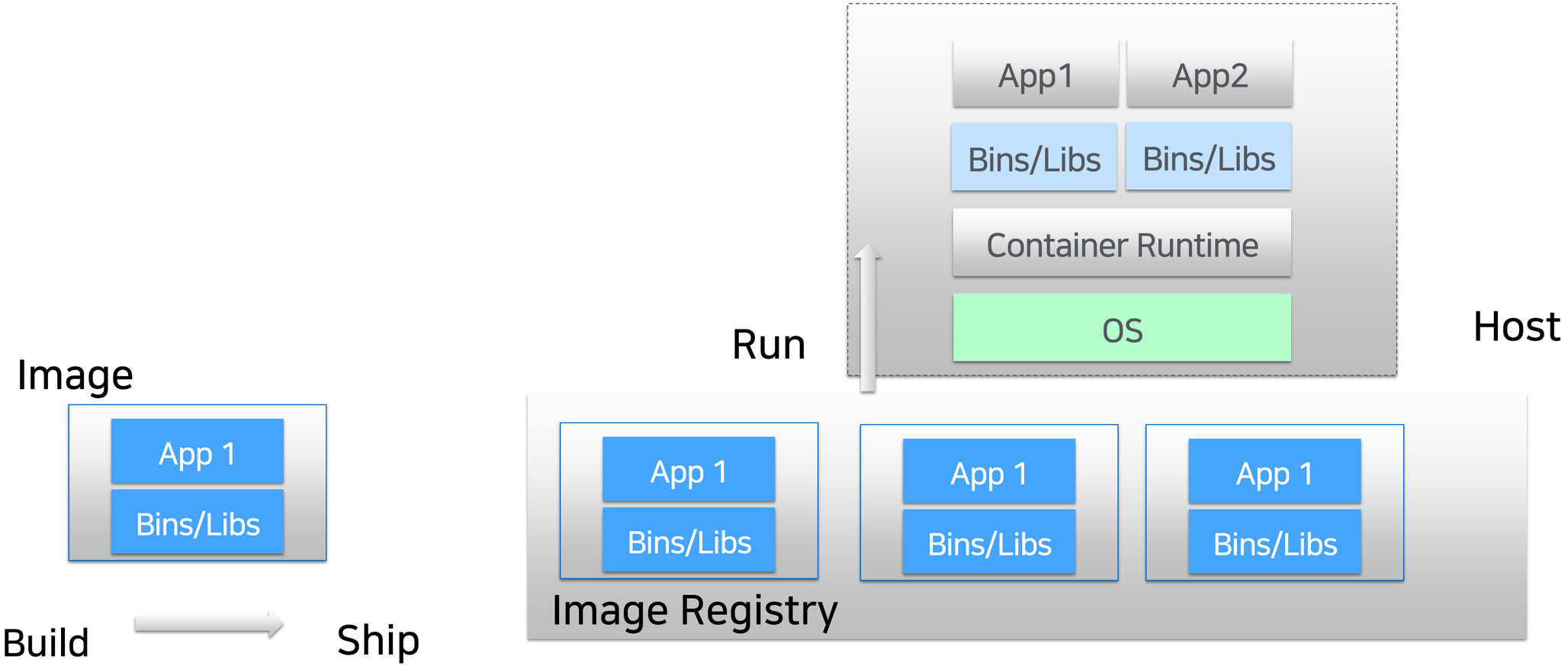
폭넓은 구동 환경

- Linux, Windows, Mac 운영체제, 가상 머신, 베어메탈, 개발자의 컴퓨터, 데이터 센터, 온프레미스 환경, 퍼블릭 클라우드 등 사실상 어느 환경에서나 구동



1.2 컨테이너 구조

기본적으로 컨테이너 환경을 사용하는 구조를 한번 봅시다.
그럼 이제 이 구조에서 살펴볼 요소들을 보자면



1.3 살펴볼 요소

host 환경과의 단절. 격리(isolation)

Image를 통한 생성

Bins/Libs가 app/service와 함께 배포됨

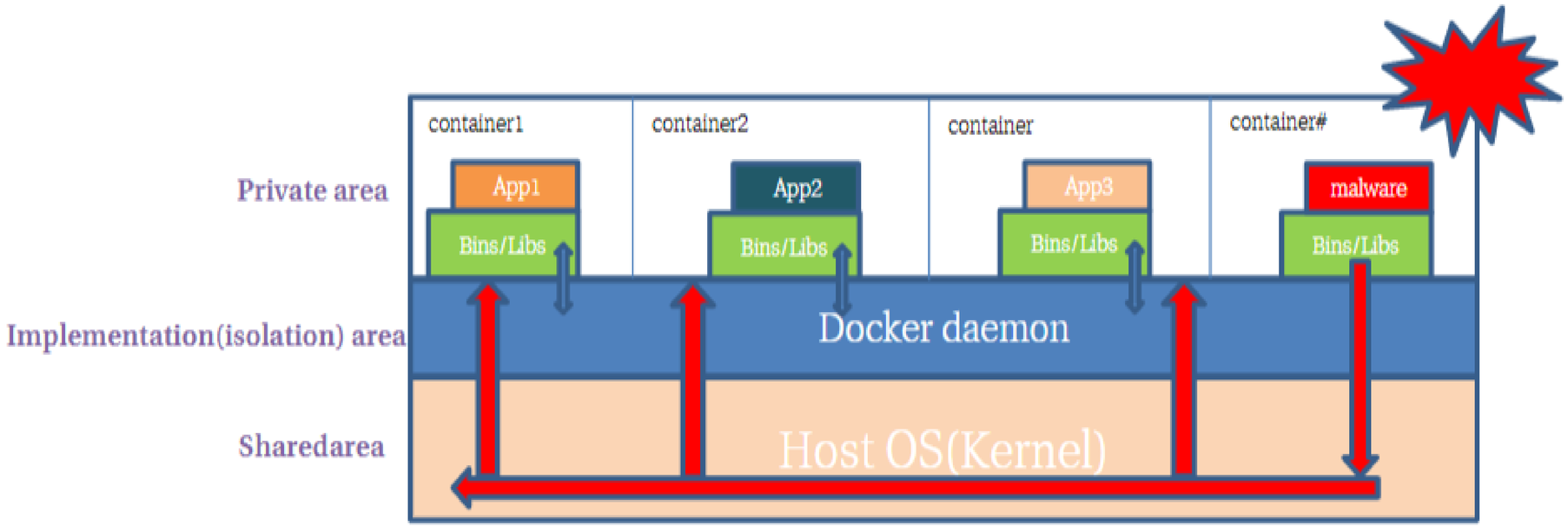
2. 컨테이너 위협 요소

2.1 Host 환경과의 단절은 깨질 수 있다.

DEVIEW 2020

container breakout

- 기본적으로 container 환경은 host와 격리된 환경
- 그러나 이는 깨질 수 있으며 이를 container breakout 이라 함.



2.1 Host 환경과의 단절은 깨질 수 있다.

container breakout이 이루어지기 위한 조건 예시

- 컨테이너 내에서 관리자(root) 권한으로 실행되어야 함.
- 컨테이너는 SYS_ADMIN Linux capability 으로 실행되어야 함
- 컨테이너에 AppArmor 프로필이 없거나 mount syscall을 허용해야 함.
- 취약한 예시: --security-opt apparmor=unconfined --cap-add=SYS_ADMIN

2.2 이미지를 통한 침해사고

DEVIEW 2020

Container 환경은 이미지를 사용하여 환경, 앱을 저장하고 배포

- 이 이미지에 악성코드, 채굴프로그램등이 심어져 있을 수 있음

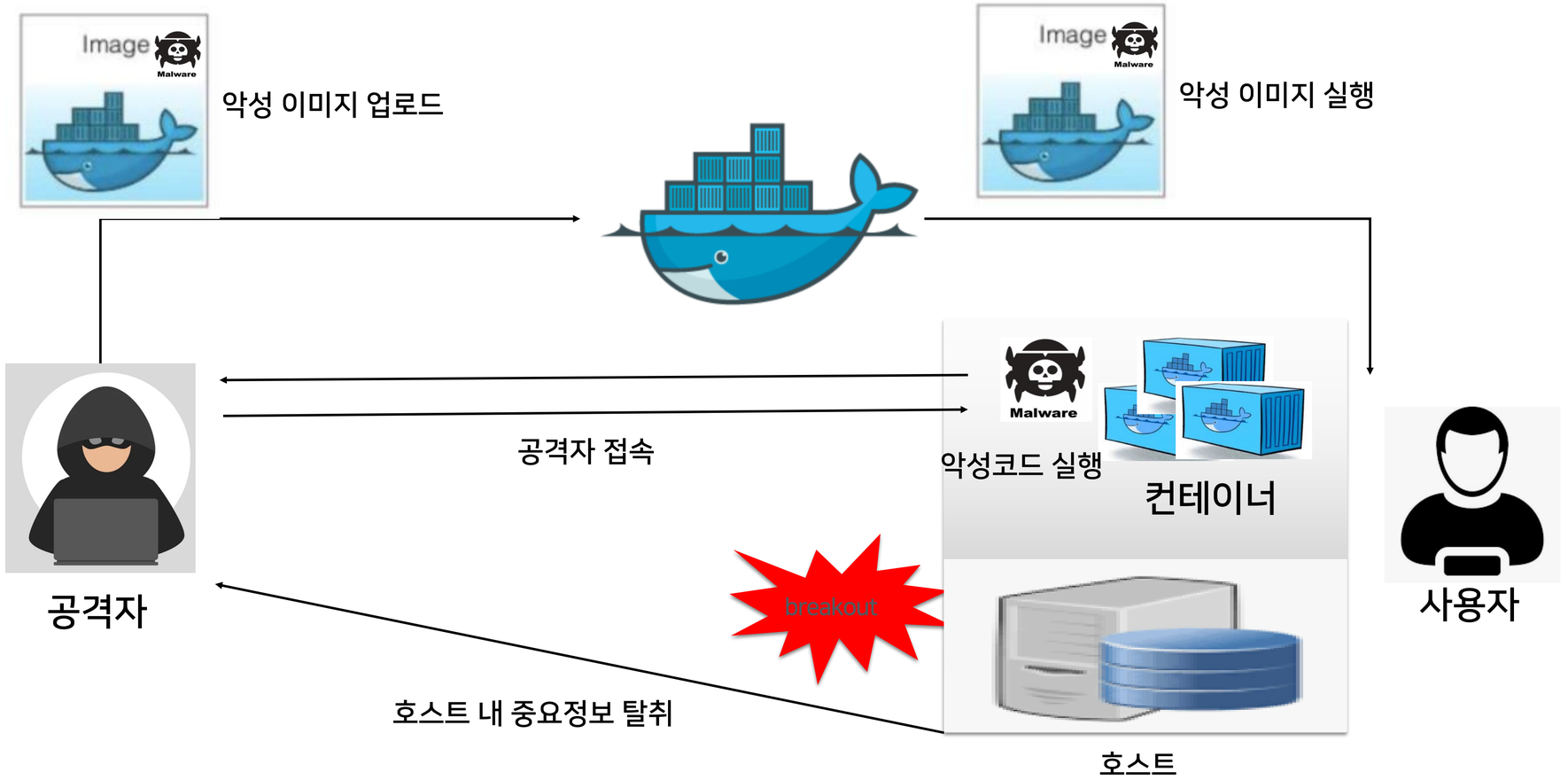
'인터넷 노출' 도커 컨테이너 노린다… 악성코드 뿌리는 ... - ITWorld
해커는 실행 시 안전하지 않은 다른 서버로 **악성코드**를 배치하는 악성 스크립트를 이용해 도커 허브로 **이미지**를 업로드했다. 연구진은 이번 공격으로 다양한 단계로 ...

www.boannews.com › media › view
도커 환경 공격하는 해커들, 전략을 또 변경했다 - 보안뉴스
2020. 7. 16. — 이 새로운 **공격** 전략의 핵심은 잘못 설정된 **도커** API 포트들을 악용한다는 것이다. 주로 컴퓨터 자원을 소모**하는** 암호화폐 채굴 **공격**에 이런 전략이 ...

Backdoored images downloaded 5 million times finally ...
2018. 6. 13. — A single person or group may have made as much as $90,000 over 10 months by spreading 17 malicious **images** that were **downloaded** more ...

2.3 침해사고 예시

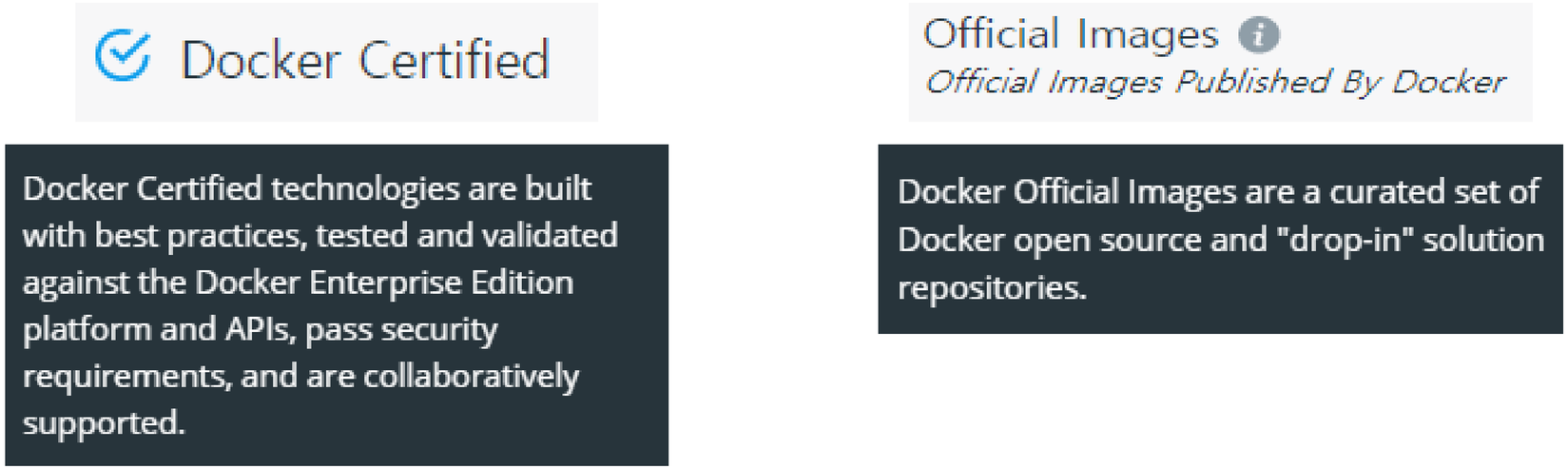
실제 침해사고 예시 시나리오를 살펴봅시다.



3. 사용자 입장에서의 개선방안

3.1 신뢰할 수 있는 이미지를 사용

- 악성 프로그램, 채굴 프로그램등을 방지하기 위해
- Docker certificate, Official image 등



3.2 필요한 권한내에서의 사용

Container 권한

- Privileged container란 컨테이너의 uid0이 호스트의 uid0과 같다는 것을 의미

옵션	설명
device 관련	해당 옵션들은 container의 격리 상태를 풀 수 있는 특수 옵션들입니다. 사용자에 의해서 추가된 옵션들에 의해 root에 버금가는 권한을 가질 수 있습니다. 이런 container가 공격자에게 노출되어 공격당하게 되면, breakout이 될 수 있습니다.
dns 관련	
--ipc, --net, --pid, --uts,userns	
--security-opt	
--sysctl	
--cap-add	
--privileged	
--user	

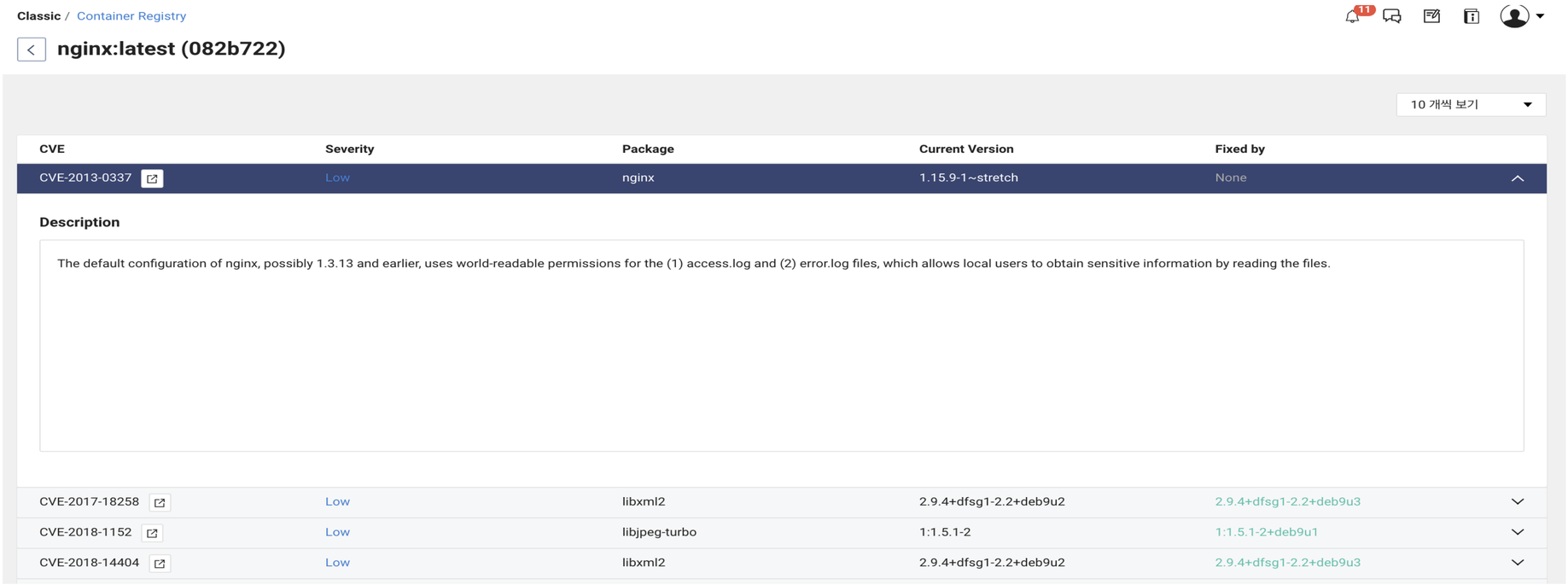
3.3 이미지 취약점 스캔 & 주기적인 업데이트

DEVIEW 2020

Cloud Provider, Major 레지스트리들의 취약점 스캔 기능을 활용

- Harbor, FlawCheck, Quay.io

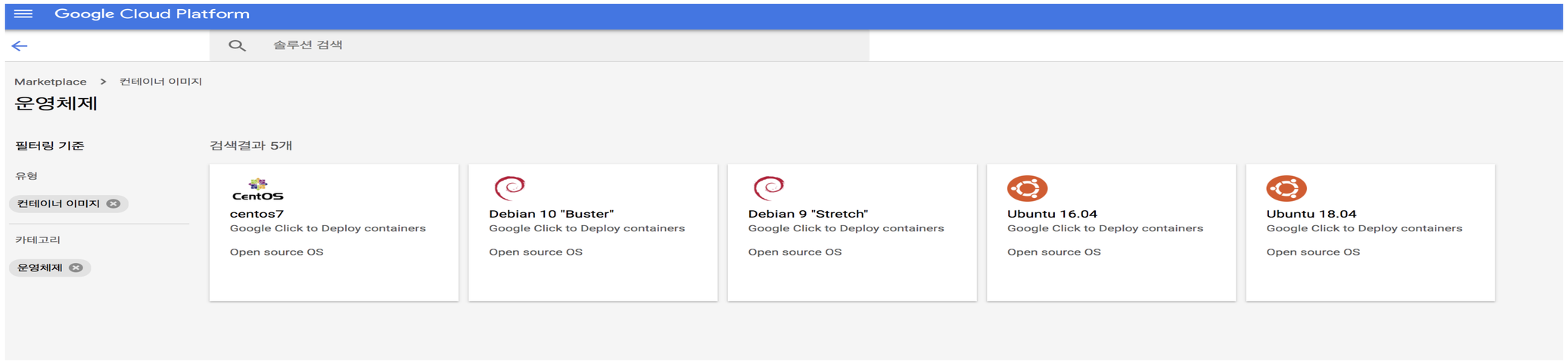
주기적인 업데이트를 통해 많은 취약점 예방이 가능

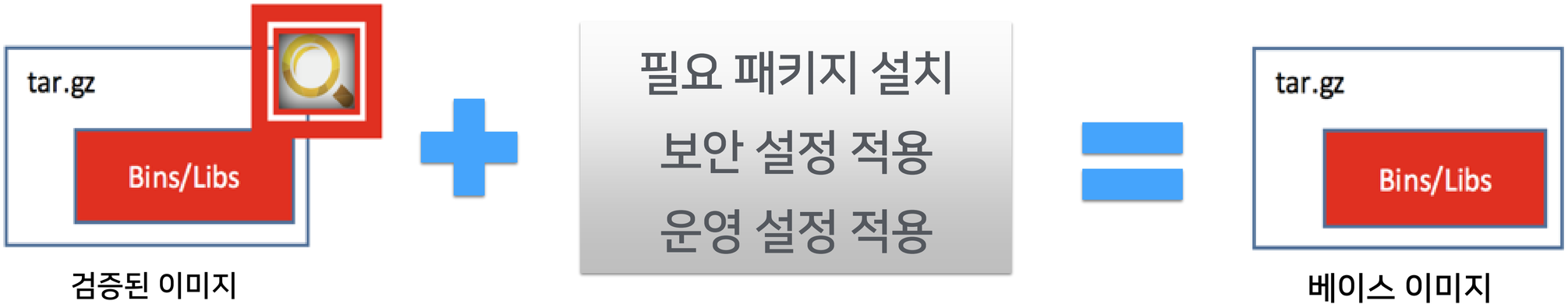


4. 플랫폼 개발자 입장에서의 개선방안

4.1 안전한 이미지 제공

- 검증된 이미지 기반으로 베이스 이미지를 생성하여 제공





검증된 이미지

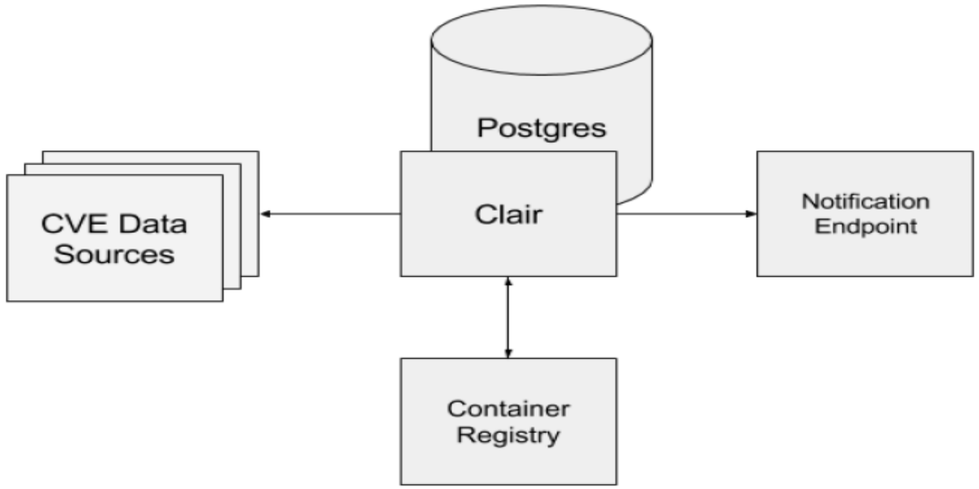
베이스 이미지

4.2 취약점 검사 제공

DEVIEW 2020

Clair

- Clair는 appc 및 docker 컨테이너 의 취약점에 대한 정적 분석을 위한 오픈 소스 프로젝트



Data Source	Data Collected	Format	License
Debian Security Bug Tracker	Debian 6, 7, 8, unstable namespaces	dpkg	Debian
Ubuntu CVE Tracker	Ubuntu 12.04, 12.10, 13.04, 14.04, 14.10, 15.04, 15.10, 16.04 namespaces	dpkg	GPLv2
Red Hat Security Data	CentOS 5, 6, 7 namespaces	rpm	CVRF
Oracle Linux Security Data	Oracle Linux 5, 6, 7 namespaces	rpm	CVRF
Amazon Linux Security Advisories	Amazon Linux 2018.03, 2 namespaces	rpm	MIT-0
SUSE OVAL Descriptions	openSUSE, SUSE Linux Enterprise namespaces	rpm	CC-BY-NC-4.0
Alpine SecDB	Alpine 3.3, Alpine 3.4, Alpine 3.5 namespaces	apk	MIT
NIST NVD	Generic Vulnerability Metadata	N/A	Public Domain

4.2 취약점 검사 제공

Layer 정보 받아오기

- Container Registry로부터 취약점 검사를 수행할 이미지의 layer 정보 받아오기
- 토큰 얻는 법: https://docs.docker.com/registry/spec/auth/token/

예제 요청

```
GET https://(registryurl)/v2/(registryname)/(imageName)/manifests/(reference)
```

```
Authorization : Basic base64 (아이디 : 패스워드)
Accept : application/vnd.docker.distribution.manifest.v2+json, application/vnd.docker.distribution.manifest.v1+prettyjws
```

4.2 취약점 검사 제공

DEVIEW 2020

예제 응답


```
{
   "schemaVersion": 2,
   "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
   "config": {
      "mediaType": "application/vnd.docker.container.image.v1+json",
      "size": 6021,
      "digest": "sha256:8c9ca4d17702c354fa41432be278d8ce6c0761b1302608034fa3ad49c6da43f9"
   },
   "layers": [
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 22500288,
         "digest": "sha256:6ae821421a7debccb4151f7a50dc8ec0317674429bec0f275402d697047a8e96"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 22262095,
         "digest": "sha256:58702d4af1973351b67463938c69fa372b9d4232218102580433762ec082544a"
      },
      {
         "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
         "size": 204,
         "digest": "sha256:b165f42e8fd4472a51e153610e90131922860647296eb604bc7b50d0547b59b5"
      }
   ]
}
```

4.2 취약점 검사 제공

DEVIEW 2020

Layer 정보 생성

- 가장 마지막 layer까지의 정보를 생성

예제 요청

```
POST http://localhost:6060/v1/layers HTTP/1.1
```


```
{
  "Layer": {
    "Name": "523ef1d23f222195488575f52a39c729c76a8c5630c9a194139cb246fb212da6",
    "Path": "https://mystorage.com/layers/523ef1d23f222195488575f52a39c729c76a8c5630c9a194139cb246fb212da6/layer.tar",
    "Headers": {
      "Authorization": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.EkN-DOsnsuRjRO6BxXemmJDm3HbxrbRzXglbN2S4sOkopdU4IsDxTI8jO19W_A4K8ZPJijNLis4EZsHeY559a4DFOd50_OqgHGuERTqYZyuhtF39yxJPAjUESwxk2J5k_4zM3O-vtd1Ghyo4IbqKKSy6J9mTniYJPenn5-HIirE"
    },
    "ParentName": "140f9bdfeb9784cf8730e9dab5dd12fbd704151cf555ac8cae650451794e5ac2",
    "Format": "Docker"
  }
}
```

4.2 취약점 검사 제공

DEVIEW 2020

예제 응답

```
HTTP/1.1 201 Created
Content-Type: application/json;charset=utf-8
Server: clair
```


```
{
  "Layer": {
    "Name": "523ef1d23f222195488575f52a39c729c76a8c5630c9a194139cb246fb212da6",
    "Path": "https://mystorage.com/layers/523ef1d23f222195488575f52a39c729c76a8c5630c9a194139cb246fb212da6/layer.tar",
    "Headers": {
      "Authorization": "Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiYWRtaW4iOnRydWV9.EkN-DOsnsuRjRO6BxXemmJDm3HbxrbRzXglbN2S4sOkopdU4IsDxTI8jO19W_A4K8ZPJijNLis4EZsHeY559a4DFOd50_OqgHGuERTqYZyuhtF39yxJPAjUESwxk2J5k_4zM3O-vtd1Ghyo4IbqKKSy6J9mTniYJPenn5-HIirE"
    },
    "ParentName": "140f9bdfeb9784cf8730e9dab5dd12fbd704151cf555ac8cae650451794e5ac2",
    "Format": "Docker",
    "IndexedByVersion": 1
  }
}
```

4.2 취약점 검사 제공

DEVIEW 2020

취약점 정보 호출

- 세 개의 레이어 A-> B-> C로 구성된 이미지의 경우 세 번째 레이어 (C)에서 정보로 호출하면 상위 레이어 (A, B)에서 수집 된 분석 데이터를 포함하여 해당 이미지에 대한 모든 취약점 정보를 반환

예제 요청

```
GET http://localhost:6060/v1/layers/17675ec01494d651e1ccf81dc9cf63959ebfeed4f978fddb1666b6ead008ed52?features&vulnerabilities
HTTP/1.1
```

4.2 취약점 검사 제공

DEVIEW 2020

예제 응답

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=utf-8
Server: clair
```


```
{
  "Layer": {
    "Name": "17675ec01494d651e1ccf81dc9cf63959ebfeed4f978fddb1666b6ead008ed52",
    "NamespaceName": "debian:8",
    "ParentName": "140f9bdfeb9784cf8730e9dab5dd12fbd704151cf555ac8cae650451794e5ac2",
    "IndexedByVersion": 1,
    "Features": [
      {
        "Name": "coreutils",
        "NamespaceName": "debian:8",
        "Version": "8.23-4",
        "Vulnerabilities": [
          {
            "Name": "CVE-2014-9471",
            "NamespaceName": "debian:8",
            "Description": "The parse_datetime function in GNU coreutils allows remote attackers to cause a denial of service
(crash) or possibly execute arbitrary code via a crafted date string, as demonstrated by the \"--date=TZ=\"123\"345\" @1\" str
ing to the touch or date command.",
            "Link": "https://security-tracker.debian.org/tracker/CVE-2014-9471",
            "Severity": "Low",
            "FixedBy": "9.23-5"
          }
        ]
      }
    ]
  }
}
```

4.3 권장 설정 만들기

DEVIEW 2020

Kubernetes

	제목	확인 방법	조치 방안	필수/선택	default 값	설명
1.1.1	익명 리퀘스트 방지	master node에서 "ps -ef | grep kube-apiserver"	--anonymous-auth=FALSE 임을 확인	필수	익명 접근을 허가함.	API 서버에 익명 리퀘스트 불가능하게 함
1.1.2	basic 인증 방지	master node에서 "ps -ef | grep kube-apiserver"	--basic-auth-file 있어서는 안됨.	필수	basisc 인증 불가	편의에 의해 많이 쓰고 있음. 다른 대체 방안이 있으니 이건 써서는 안됨
1.1.3	insecure 토큰 방지	master node에서 "ps -ef | grep kube-apiserver"	--insecure-allow-any-token 이 있어서는 안됨	필수	불안전한 토큰 불가	인증 없는 토큰은 무시하여야함.
1.1.4	https 통신 확인	master node에서 "ps -ef | grep kube-apiserver"	--kubelet-https 이 없거나 true로 세팅 되어있어야함.	필수	https	기본이 https임.
1.1.5	api server bind address 금지	master node에서 "ps -ef | grep kube-apiserver"	--insecure-bind-address 이 없거나 127.0.0.1로 세팅되어 있어야함	필수	127.0.0.1	누구나 API server에 쉽게 접근해서는 안됨. 만약 인증 인가가 잘 설정되어있다면 논외
1.1.6	insecure api server port open 금지	master node에서 "ps -ef | grep kube-apiserver"	--insecure-port=0 임을 확인	필수	8080	insecure port는 기본 값은 8080으로 되어있음.
1.1.7	secure api server port open	master node에서 "ps -ef | grep kube-apiserver"	--secure-port값이 1~65536 사이로 세팅 되어 있어야함.	필수	6443	secore port는 기본 값이 6443으로 되어있음.

docker

	제목	확인 방법	조치 방안	필수/선택	default 값	설명
1.1.1	호스트 OS 접근 제어	format '{{ .Id }}: Volumes={{ .Mounts }}'	주요 시스템 디렉토리 마운트 금지(/boot, /dev, /etc, /lib 등등)	필수		컨테이너에서 호스트OS 주요 자원에 대한 수정으로 안정성과 보안에 영향 미칠 수 있음.
1.1.2	네임스페이스 관리	docker ps --quiet --all | xargs docker inspect --fo	--net=host, --pid=host, --ipc=host,--uts=host 사용 금지	필수		권한 상승 공격을 방지하기 위해 컨테이너가 불필요하게 과도한 권한을 가지지 않도록 함
1.1.3	SSL/TLS 적용	ps -ef | grep dockerd	docker --tlsverify --tlscacert=ca.pem --tlscert=cert.pem --tlskey=key.pem -H=$HOST:2376 version'	필수		데이터 스니핑으로부터의 보호를 위해 설정 필요

4.4 사용자 별 권한 설정

DEVIEW 2020

사용자 별 권한

- 사용자 별로 필요한 권한내에서만 사용할 수 있도록 권한을 분리

역할	역할이름	설명
admin	총 관리자	클러스터 및 관련 kubernetes api 객체를 전체 관리할 수 있는 권한을 가짐
cluster admin	클러스터 관리자	클러스터 관리에 대한 액세스 권한을 가지며, 일부 설정이나 secret등에 대한 접근은 제한 될 수 있음.
cluster member	클러스터 멤버	클러스터를 가져오고 볼 수 있는 권한을 가짐
serviceuser	서비스 사용자	리소스에 대한 읽기 전용액세스 권한을 가짐

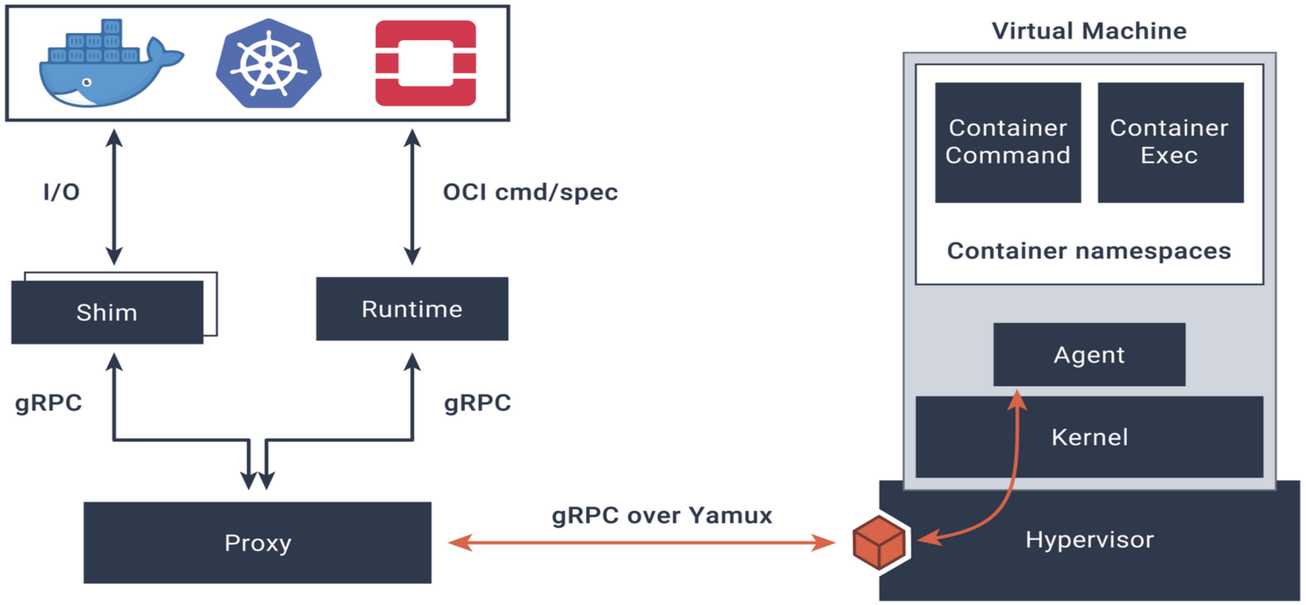
4.5 런타임 보호

옵션을 통한 보호

- docker(Capabilities 제한 & AppArmor, SELinux, seccomp & userremap)

보안이 강화된 런타임 사용

- rkt, gvisor, kata container 등등

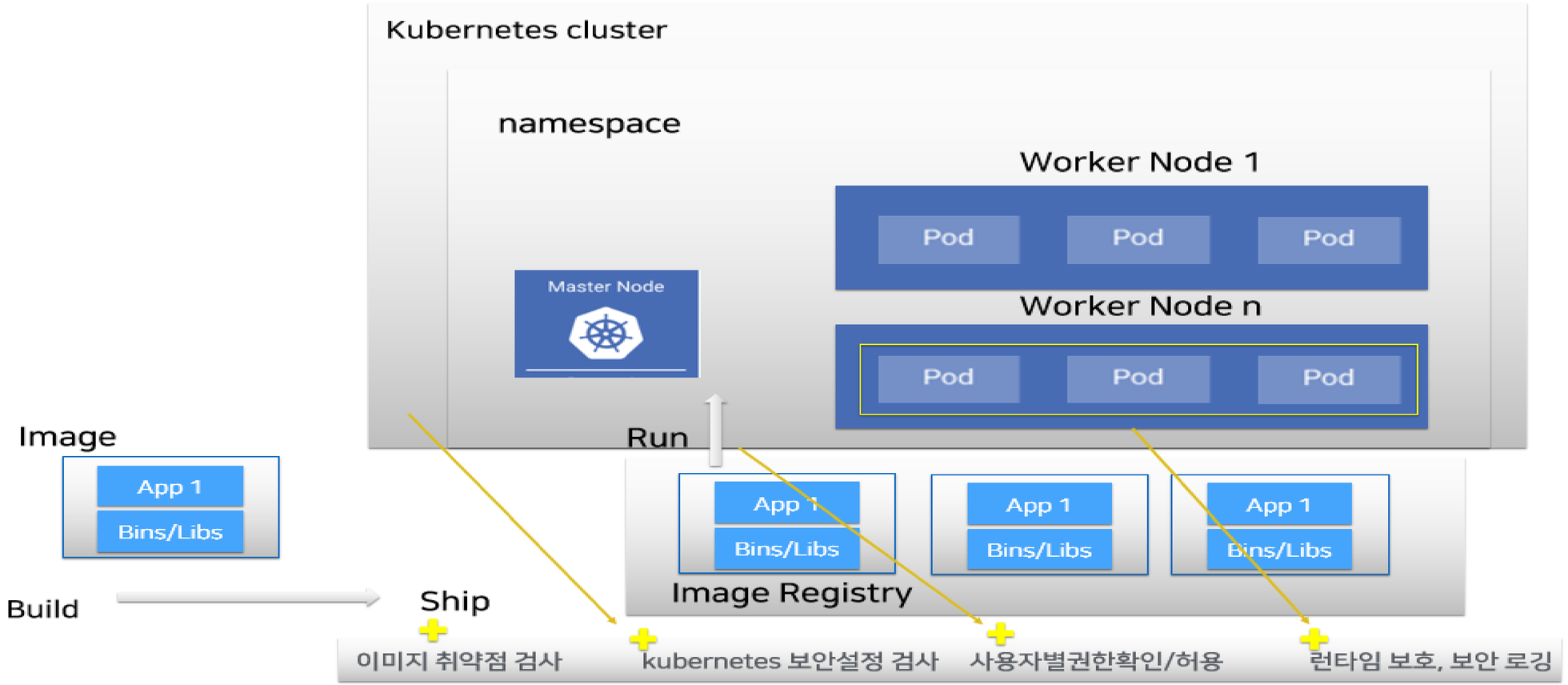


4.5 로깅

- 로그 기록을 정기적으로 확인/감독하여 오류 및 부정행위가 발생하거나 예상되는 경우 즉각적인 확인, 조치가 가능하도록 하는 것이 좋음

4.6 통합 예시

- 이미지 올라올 시: 취약점 스캐닝
- 사용자 별 권한 기반으로 작업 허용
- 컨테이너 방화벽, 런타임 설정 등을 통한 보호



Q & A

Thank You