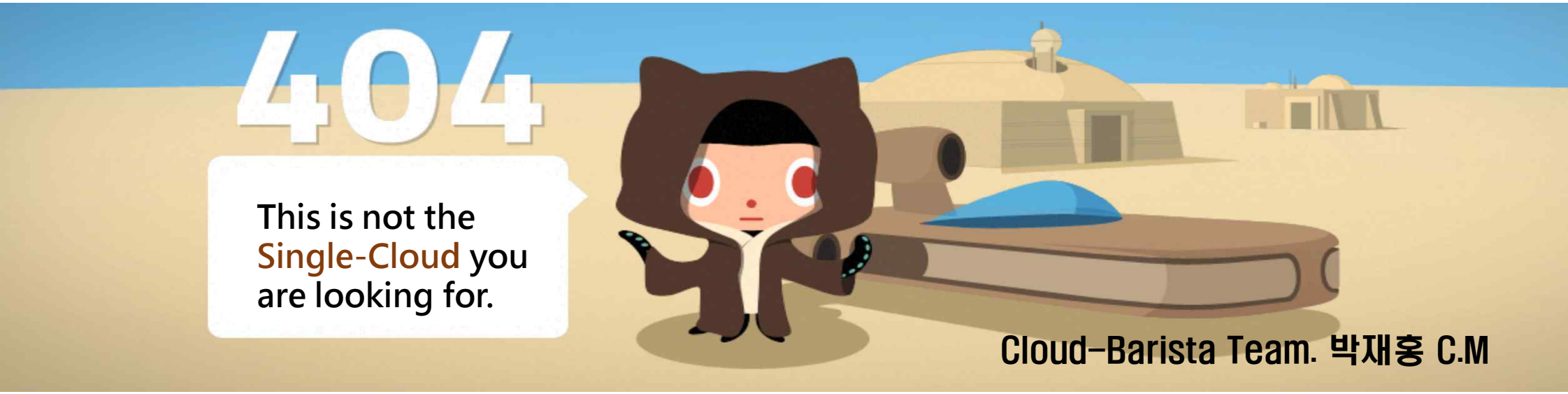
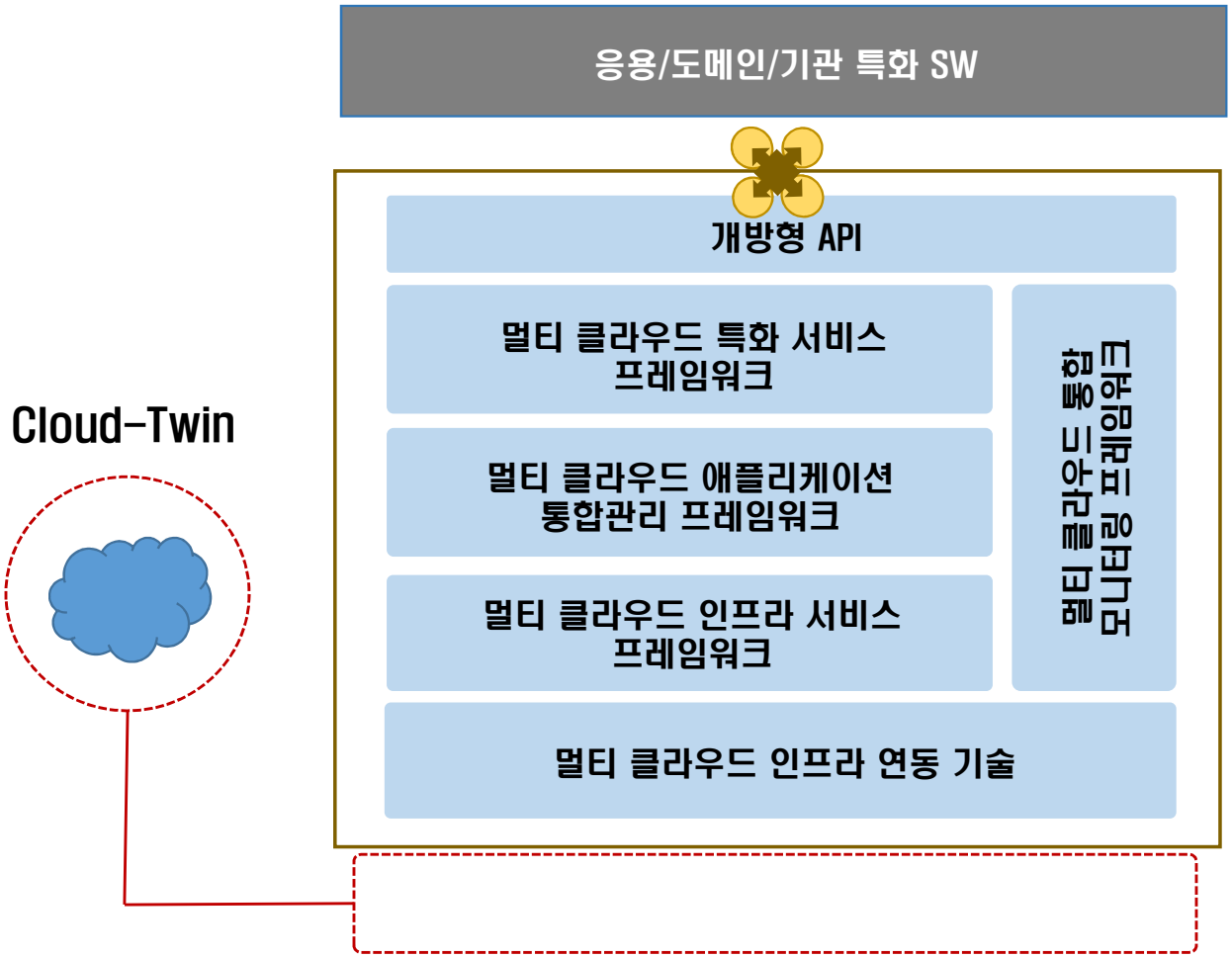
CLOUD BARISTA 멀티 클라우드 서비스 공통 프레임워크

멀티 클라우드 인프라 에뮬레이터
(Cloud-Twin)



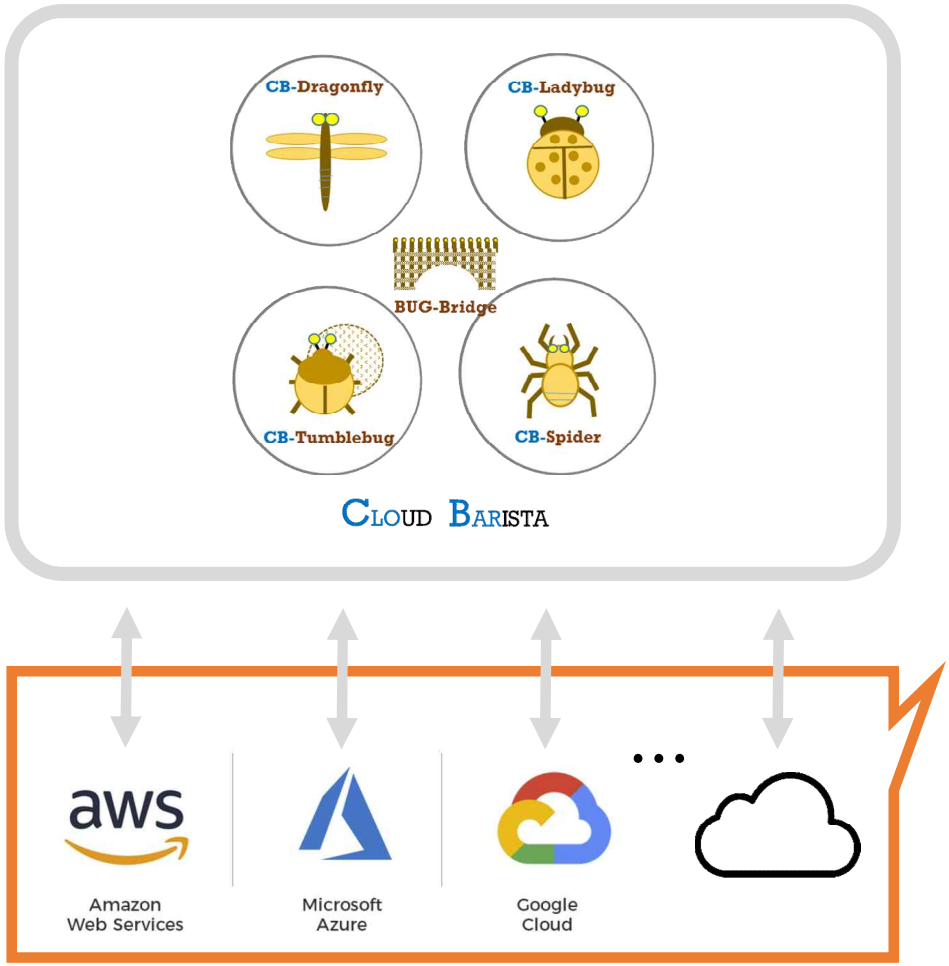
Cloud-Barista Team. 박재홍 C.M

이번 세션은…


응용/도메인/기관 특화 SW
개방형 API
멀티 클라우드 특화 서비스 프레임워크
멀티 클라우드 애플리케이션 통합관리 프레임워크
멀티 클라우드 인프라 서비스 프레임워크
멀티 클라우드 인프라 연동 기술
멀티 클라우드 플랫폼 통합관리 프레임워크
Cloud-Twin

목 차

필요성



Cloud-Barista 시스템 개발시 다종, 다수의 클라우드 서비스 필요

하지만,

#1 개발자 마다 대규모의 멀티 클라우드 개발 환경을 꾸미기 어려움

- 다종, 다수의 클라우드 생성 및 환경 설정 필요 (피곤)
- 클라우드 서비스 이용료 발생 등

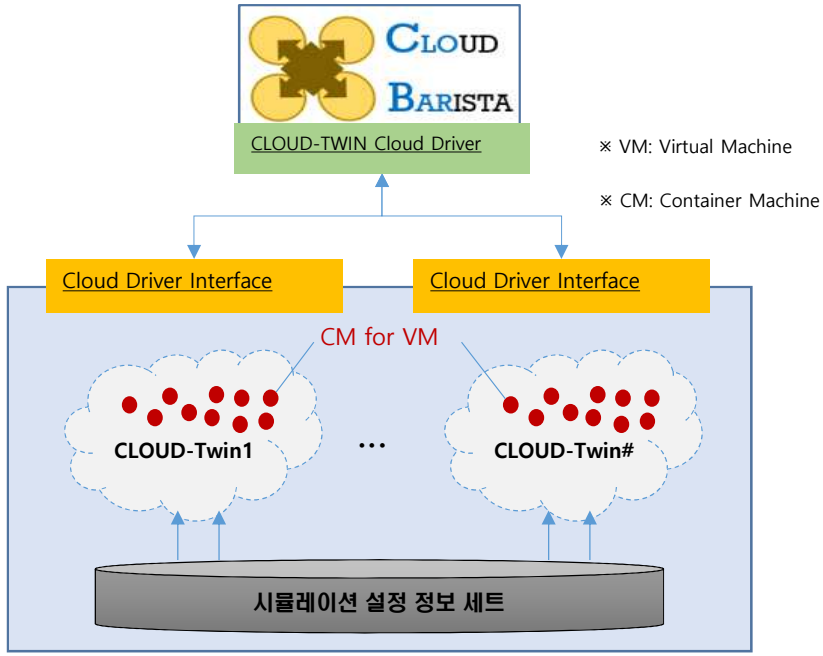
#2 개발환경에서 클라우드 서비스 예외 상황을 시험하기 어려움

- 상용 클라우드 시스템에 임의로 예외 발생 할 수 없음.
- 예)
  - vm 비정상종료 재현 불가
  - 네트워크 지연, 장애 재현 불가
  - 기타 기능별 다양한 장애 발생 재현 불가
  - ...

Cloud-Twin 시스템 목적

(목적) 멀티 클라우드 기반의 SW 개발 및 검증 편의성을 제공하기 위하여, 소규모의 시스템으로 대규모 가상 멀티 클라우드 환경을 제공하는 클라우드 인프라 에뮬레이션 기술 개발

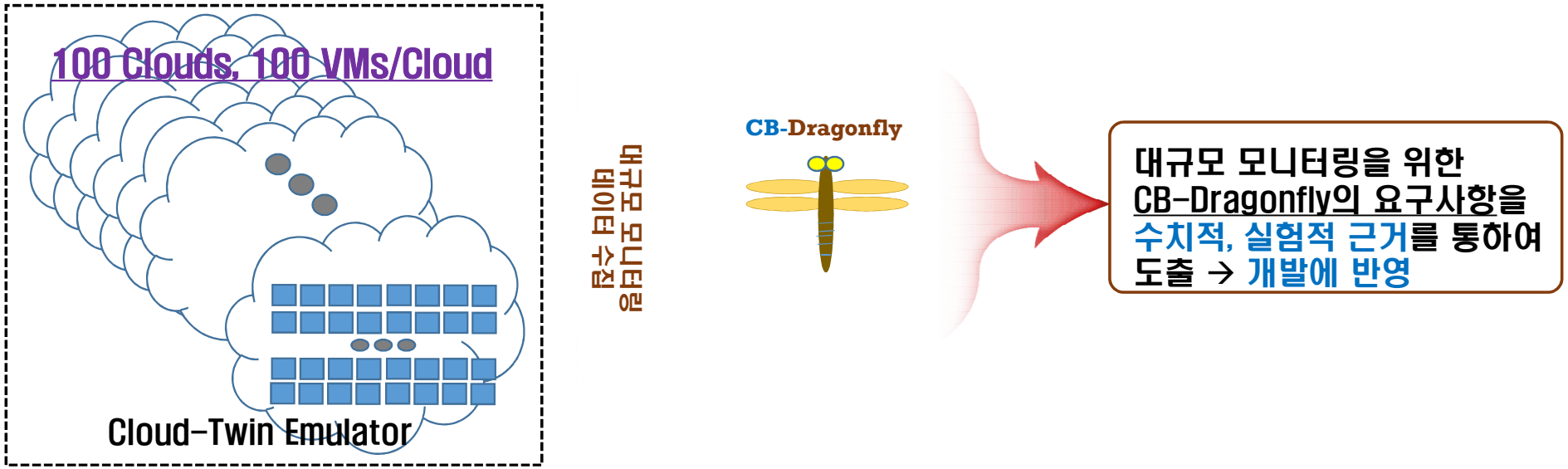
- 다양한 CSP 사업자의 클라우드 연동을 위한 난관(예산, 시간, 노하우 등)을 해결
- 멀티 클라우드 환경하에서의 다양한 상황(장애, 성능, 기능 등)을 손쉽게 설정할 수 있는 수단 제공



<CLOUD-TWIN 시스템 개략 구조>

Cloud-Twin 활용 예시

- 대규모의 멀티 클라우드 환경에서의 기술 검증(PoC)을 위한 개발 환경
  - Cloud-Twin을 통한 100개의 클라우드 생성, 클라우드당 100개의 VM을 배치
  - 각 VM에 CB-Agent 설치, CB-Agent는 구동시 Cloud-Barista에 접속정보 등록
  - CB-Dragonfly에서 모니터링 Metric 기반의 주기적 모니터링 수행(e.g. 주기 : 5초)
  - 대규모 모니터링 환경을 위한 CB-Dragonfly의 한계점(Issue)들 발굴
    - 저장소 입출력 성능, 모니터링 서버의 성능, 네트워크 지연, Cloud-Barista의 전체 성능 등



[클라우드 에뮬레이터의 활용 사례]

Cloud-Twin 시스템의 구현 개요

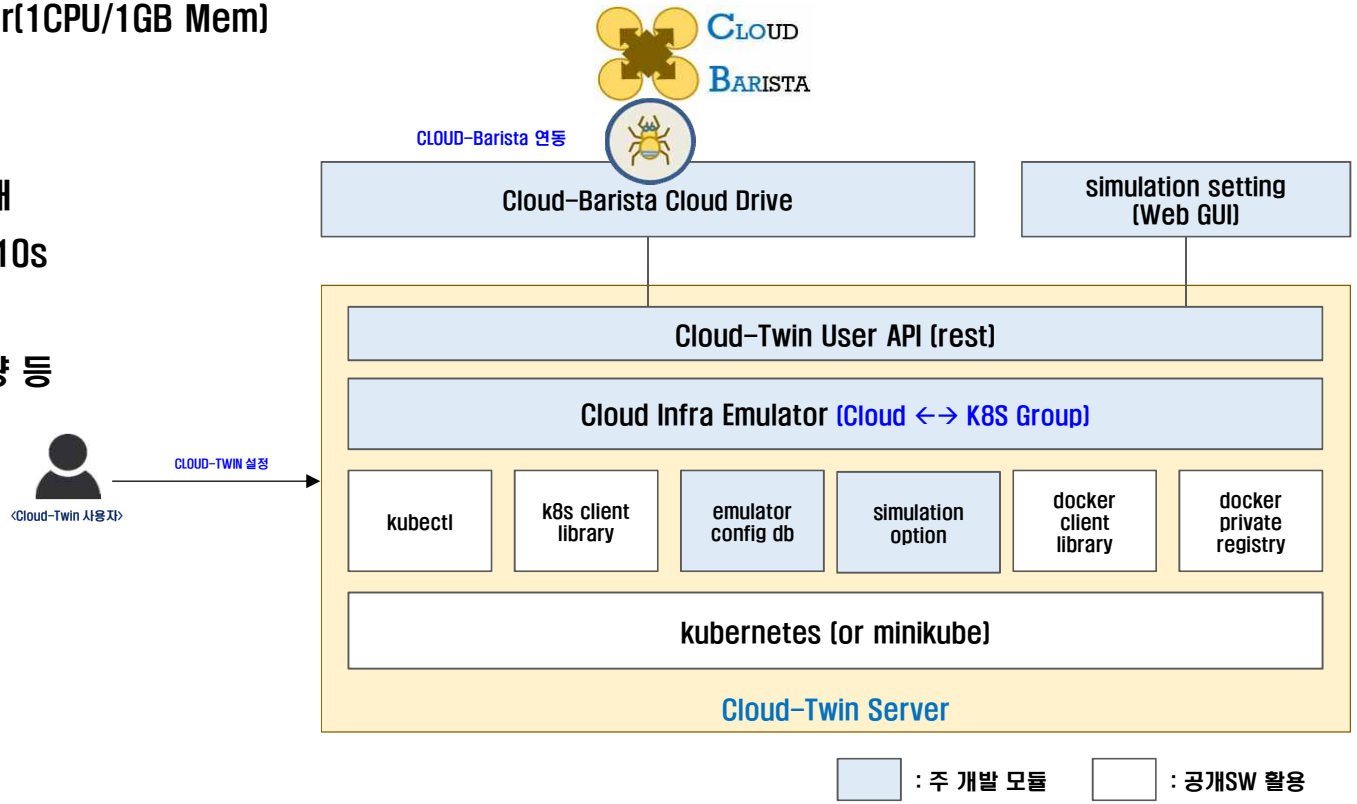
컨테이너 기반 클라우드 서비스 에뮬레이션 (+시뮬레이션)

K8S 기반 Container로 VM 동작을 매핑

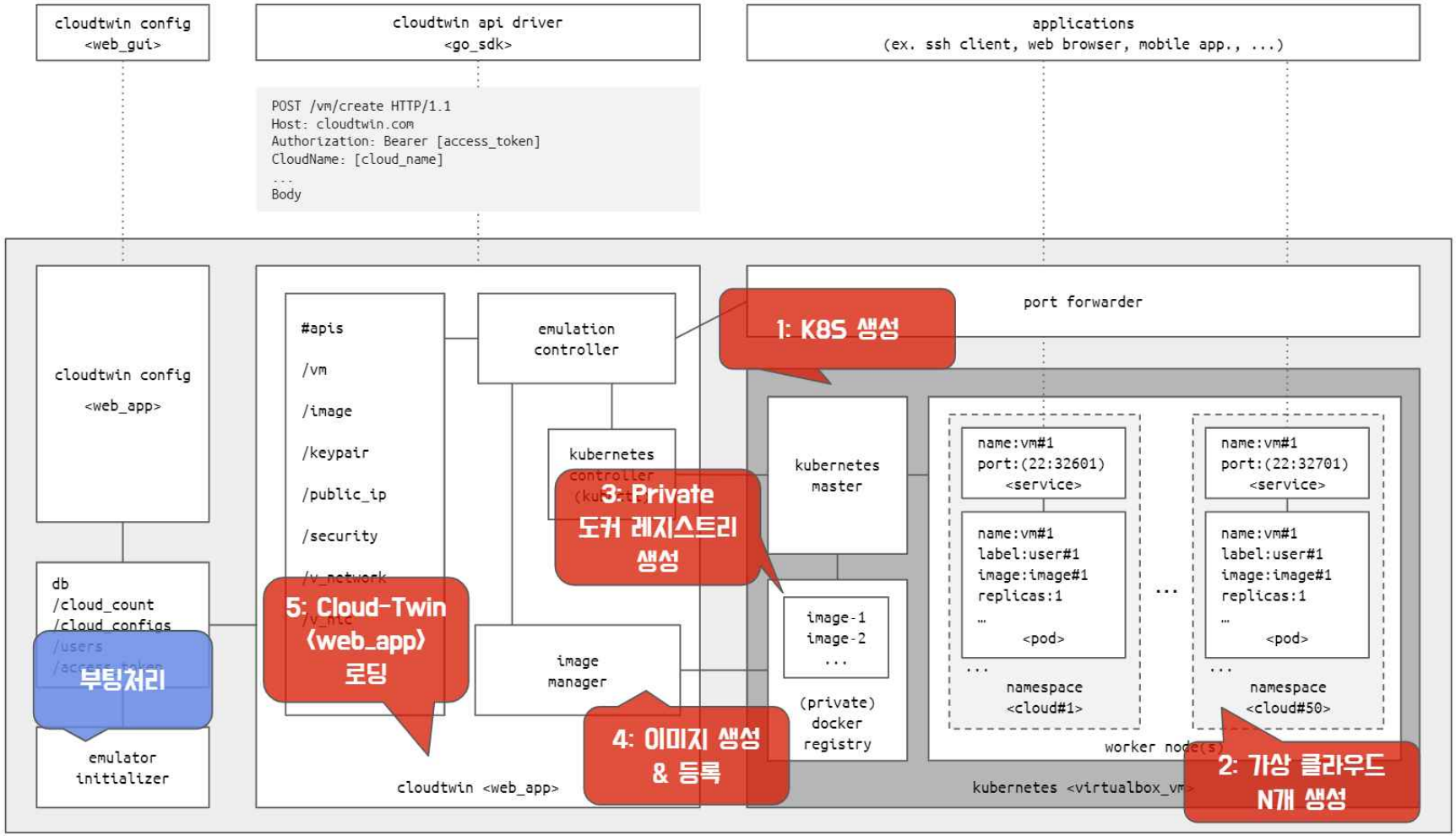
- VM(1CPU/1GB Mem) : Container(1CPU/1GB Mem)

클라우드 서비스 에뮬레이션 설정

- 기능별 장애발생 확률 설정
  - VM 생성시 0.2 확률로 실패
  - 네트워크 지연시간은 5s~10s
- VM 상태 정보 에뮬레이션
  - cpu, network, disk 사용량 등



Cloud-Twin 동작: Booting 1/2



Cloud-Twin 동작: Booting 2/2

1.kubernetes 실행
./minikube start --memory=8192 --cpus=4 --disk-size=10g

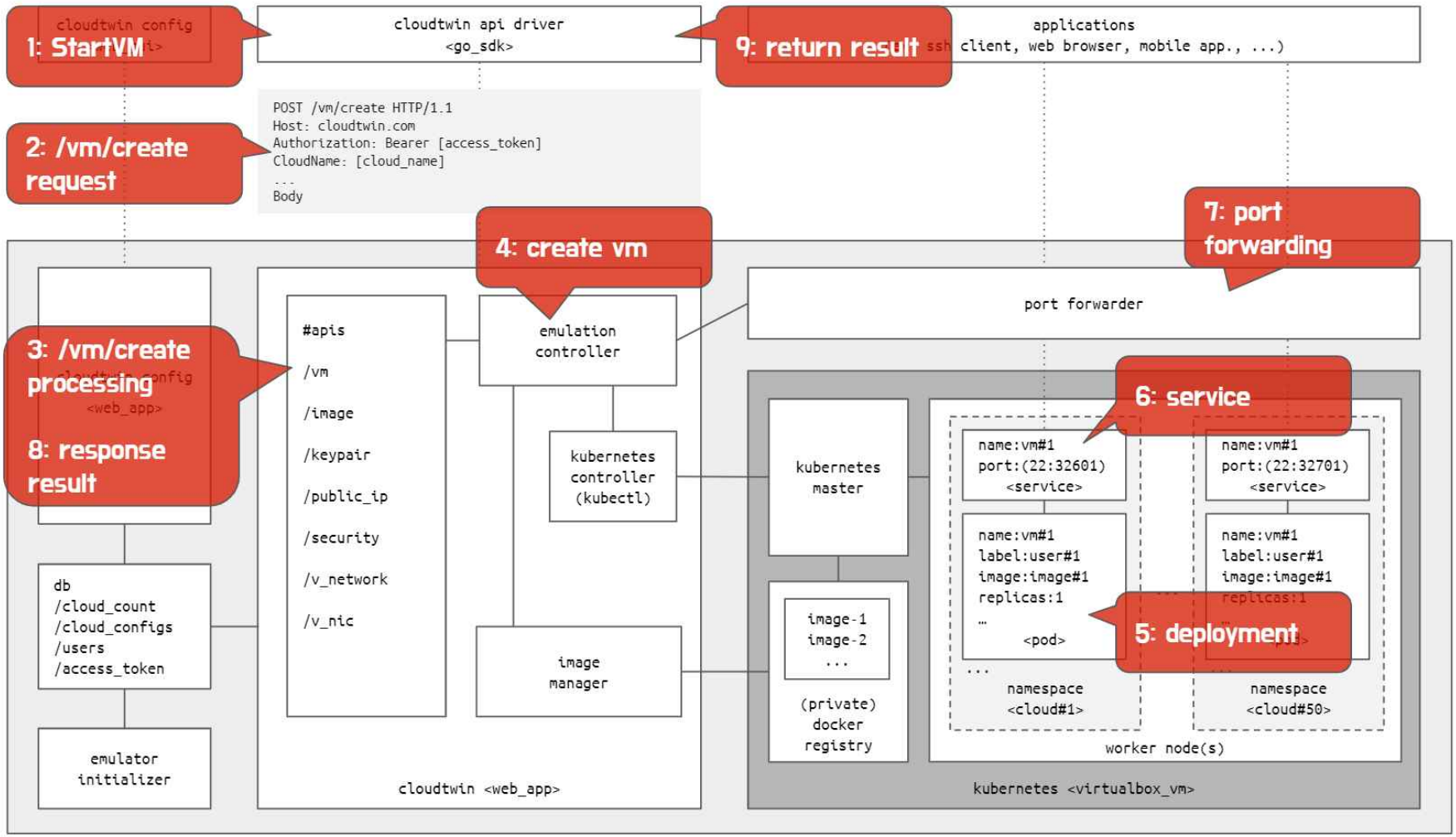
2.가상 cloud 생성
클라우드 개수 설정 정보에 따라서 namespace 생성 (ex. cloud-[1,2,3...50])
'kubectl create namespace cloud-{} -o json'.format(i + 1)

3.(private) docker registry 설치 & 실행
sudo docker run -d -p 5000:5000 --name registry registry:2
# docker private registry 설정
# /etc/docker/daemon.json
{ "insecure-registries" : ["192.168.99.100:5000"] }

4.image 생성&등록 docker 이미지 생성 & 등록 (이미지 등록 안되어 있는 경우에만 실행)
# 도커 이미지 등록 (도커이미지 생성과정 생략)
sudo docker push 192.168.99.100:5000/ssh-server
# 조회
curl http://192.168.99.100:5000/v2/_catalog
{"repositories":["ssh-server"]}

5.cloudtwin <web_app> 로딩
port forwarder 로딩
cloudtwin config <web_app> 로딩

Cloud-Twin 동작: VM Create 1/4



Cloud-Twin 동작: VM Create 2/4

1.애플리케이션이 cloudtwin driver의 VMHandler.StartVM 기능을 호출한다.

```
// create a driver and connect
driver := new(cloudtwin.Driver)
connectionInfo := ifs.ConnectionInfo{}
connectionInfo.CredentialInfo.IdentityEndpoint = "http://192.168.0.1:80"
connectionInfo.CredentialInfo.DomainName = "cloud-1"
connection, _ := driver.ConnectCloud(connectionInfo)

vmHandler, err := connection.CreateVMHandler()
if err != nil {
   panic(err)
}

vmReqInfo := irs.VMReqInfo{
   Name: "vm-1",
   ImageInfo: irs.ImageInfo{
      Id: "image-1",
   },
   SpecID: "spec-1",
   //...
}
vmInfo, err := vmHandler.StartVM(vmReqInfo)
```

2. VMHandler.StartVM은 cloudtwin에게 /vm/create 메시지를 생성하고 전송한다.

```
POST /vm/create HTTP/1.1
Host: localhost:5000
Authorization: Bearer access_token
CloudName: cloud-1
Content-Type: application/json
//...

{
   "vm_name": "vm-1",
   "image_id" : "image-1",
   "spec_id": "spec-1",
   ...
}
```

Cloud-Twin 동작: VM Create 3/4

3. cloudtwin은 요청 메시지를 분석해서 해당 명령을 호출한다. 본 과정에서는 vm/create 명령을 호출하다.
4. emulation controller는 kubernetes controller에게 vm 생성을 위한 일련의 명령 호출을 요청한다.
5. kubernetes controller는 shell 명령으로 deployment와 service 생성을 요청한다. 이때 부가정보는 kubernetes object의 label에 기록한다.
6. kubernetes master는 deployment를 생성명령을 호출한다. 그리고 이에 대한 service 생성명령을 호출한다.

```
# deployment 명령 요청 & 결과의 예)
kubectl create deployment vm-1 --namespace=cloud-1 --image localhost:5000/ssh-server -o json
# label 기록 요청 & 결과의 예)
kubectl label deployment vm-1 --namespace=cloud-1 user=user-1 uid=7cd2e0ae-ec91-4ea1-b939-b4b1f36f552f type=spec-1 -o json
# service 생성 명령 요청 & 결과의 예)
kubectl expose deployment vm-1 --port=22 --type=LoadBalancer --namespace=cloud-1 -o json
```

7. kubernetes worker node는 deployment에 해당하는 pod를 생성한다.
8. kubernetes worker node는 deployment에 대응되는 service를 생성한다.

9. emulation manager는 등록된 service에 접근가능한 port번호를 조회하고, 이를 port forwarder에 등록한다.
port 번호는 service 생성 결과로 반환된 정보의 “spec/ports[0]/nodePort” 정보를 사용한다.

Cloud-Twin 동작: VM Create 4/4

10. cloudtwin <web_app>은 vm 생성정보를 결과를 반환한다. (성공, 실패)

- 성공 결과의 예)

```
msg = {
    'err': None,
    'res': {
        'vm_name': 'vm-1',
        'vm_id': '7cd2e0ae-ec91-4ea1-b939-b4b1f36f552f',
        'start_time': '2019-09-18T04:52:03Z',
        'etc': {
            'ssh_port': '31065'
        }
    }
}
```

- 실패 결과의 예)

```
msg = {
    'err': 'Error from server (AlreadyExists): deployments.apps "vm-1" already exists',
    'res': None
}
```

11. cloudtwin driver는 cloudtwin <web_app>가 전달한 결과를 파싱해서 반환한다.

로드맵

Americano

Cloud Driver 개발

- 현황: Lifecycle 기본 동작 개발
- 계획: 개정 driver API 통합 시험 및 반영 예정

Cloud-Twin Server 개발

- 현황: 핵심 기능 개발
- 계획: 시뮬레이션 기능 및 Cloud-Barista 통합 시험을 통한 개선 예정

Espresso

Cloud-Twin 확장 개발

- 현황: Americano 기본 기능
- 계획: 활용 후 필요 기능 추가 및 보완 개발 예정

- 예)
  - 자원 및 기능 에뮬레이션 기능 추가
  - 관리 도구, GUI 기능 추가 등

Demo: 대규모 멀티 클라우드 에뮬레이션

감사합니다.

www.github.com/cloud-barista

(박재홍 / jaehong@mycq.co.kr)