

개방형 클라우드 플랫폼 기술지원 및 유지보수·관리

cAdvisor Guide

모니터링

|  |  |  |
| --- | --- | --- |
| 작성자 | 오한수 (인) | 2020.12.21 |
| 검토자 | 김태진 (인) | 2020.12.21 |
| 승인자 | 정기봉 (인) | 2020.12.21 |

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

개 정 이 력

| 버전 | 작성일 | 변경내용[[1]](#footnote-1) | 작성자 | 검토자 | 승인자 |
| --- | --- | --- | --- | --- | --- |
| 0.1 | 2020.10.06 | 최초작성 | 오한수 | 김태진 | 정기봉 |
| 0.2 | 2020.11.18 | cAdvisor 문서 현행화 | 오한수 | 김태진 | 정기봉 |
| 0.3 | 2020.12.21 | 기존 모니터링 소스 문서 내용 추가 | 오한수 | 김태진 | 정기봉 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**목차**

[1. Goland 설치 4](#_Toc59451582)

[2. cAdvisor 다운로드 및 환경 구축 4](#_Toc59451583)

[1) cAdvisor 다운로드 4](#_Toc59451584)

[2) cAdvisor 환경 구축 5](#_Toc59451585)

[3) cAdvisor IDE 실행 7](#_Toc59451586)

[4) cAdvisor 웹 콘솔 확인 9](#_Toc59451587)

[3. cAdvisor 빌드 10](#_Toc59451588)

[1) cAdvisor 빌드 10](#_Toc59451589)

[2) 빌드 후 바이너리 실행 10](#_Toc59451590)

[4. cAdvisor 테스트 12](#_Toc59451591)

[1) 로컬 테스트 12](#_Toc59451592)

[2) 실 서버 테스트 (cadvisor 바이너리 파일 위치 및 실행 ) 13](#_Toc59451593)

[○ paasta deployment 및 diego-cell vm 확인 13](#_Toc59451594)

[○ diego-cell 접속 13](#_Toc59451595)

[○ cadvisor 바이너리 파일 위치 14](#_Toc59451596)

[○ cadvisor 프로세스 확인 14](#_Toc59451597)

[○ cadvisor 상태 체크 API 15](#_Toc59451598)

[○ cadvisor 컨테이너 조회 API 15](#_Toc59451599)

[○ cadvisor 컨테이너 세부 조회 API (실제 garden안에 특정 container 정보 조회 내용) 16](#_Toc59451600)

[○ Influxdb 컨테이너 모니터링 정보 16](#_Toc59451601)

[○ influxdb 전체 컨테이너 모니터링 데이터 개수 확인 16](#_Toc59451602)

[○ Influxdb 컨테이너 모니터링 하루 총 개수 확인 (10월 24일) 16](#_Toc59451603)

[5. cAdvisor 수정사항 17](#_Toc59451604)

[1) 수정사항 17](#_Toc59451605)

[6. cAdvisor 실 배포 절차 17](#_Toc59451606)

[1) 과정 17](#_Toc59451607)

[7. cAdvisor 히스토리 19](#_Toc59451608)

[1) 테스트 19](#_Toc59451609)

[2) 기존 customizing\_cadvisor 소스 내용 (반영) 20](#_Toc59451610)

### **1. Goland 설치**

### **2. cAdvisor 다운로드 및 환경 구축**

#### 1) cAdvisor 다운로드

|  |
| --- |
| ○ Git 정보  - 주소 : <https://github.com/PaaS-TA/PaaS-TA-Monitoring/tree/v5.1.0-dev>)  - 버전 : V5.1.0-dev  - cadvisor path : paasta-agents > cadvisor  $ git clone -b v5.1.0-dev <https://github.com/PaaS-TA/PaaS-TA-Monitoring.git> |

#### 2) cAdvisor 환경 구축

|  |
| --- |
| ○ Goland > Open Project 선택  $ ~/PaaS-TA-Monitoring/paasta-agents/cadvisor    ○ File > Settings 선택  - Go > GOROOT 설정 (Download Go SDK를 이용.)    - Go > GOPATH 설정    - Go Sync 설정 (cadvisor > cmd > internal > storage > influxdb > influxdb.go)  Sync dependencies of github.com/google/cadvisor/cmd click |

#### 3) cAdvisor IDE 실행

|  |
| --- |
| ○ Navigation > Edit Configurations… 선택    ○ Edit Configurations 설정    ○ Edit Configurations 값  - Name : 데몬 실행 이름 (**cAdvisor**)  - Run kind : 실행 종류 (**Package**)  - Package Path : 실제 cadvisor.go 존재 경로 (**github.com/google/cadvisor/cmd**)  - Output directory : 빌드 후 바이너리 파일 경로  - Working Directory : 실제 cadvisor 최상위 경로  (**~/PaaS-TA-Monitoring/paasta-agents/cadvisor**)  - Program arguments : 프로그램에 필요한 환경 변수  (**-storage\_driver=influxdb -storage\_driver\_db=container\_metric\_db -storage\_driver\_host=127.0.0.1:8069 -allow\_dynamic\_housekeeping=true -housekeeping\_interval=30s**)  ○ OK 선택  ○ Debug 실행 및 결과 |

#### 4) cAdvisor 웹 콘솔 확인

|  |
| --- |
| ○ cAdvisor 웹 콘솔 (local) |

### **3. cAdvisor 빌드**

#### 1) cAdvisor 빌드

|  |
| --- |
| ○ Makefile 이용  $ cd ~/PaaS-TA-Monitoring/paasta-agents/cadvisor  $ make build    - make build 후 cadvisor 바이너리 파일 생성 시 과정 및 결과 |

#### 2) 빌드 후 바이너리 실행

|  |
| --- |
| ○ cadvisor 바이너리 파일 위치 및 실행  $ cd ~/PaaS-TA-Monitoring/paasta-agents/cadvisor  $ ./cadvisor -storage\_driver=influxdb -storage\_driver\_db=container\_metric\_db -storage\_driver\_host=127.0.0.1:8069 -allow\_dynamic\_housekeeping=true -housekeeping\_interval=30s |

### **4. cAdvisor 테스트**

#### 1) 로컬 테스트

|  |
| --- |
| ○ cadvisor 실행 여부 및 응답  $ curl <http://127.0.0.1:8080/healthz>    ○ cadvisor containers 정보  $ curl <http://127.0.0.1:8080/api/v1.0/containers> |

#### 2) 실 서버 테스트 (cadvisor 바이너리 파일 위치 및 실행 )

|  |
| --- |
| ○ paasta deployment 및 diego-cell vm 확인 $ bosh -e micro-bosh vms   ○ diego-cell 접속 $ bosh -e micro-bosh -d paasta ssh diego-cell/11c87e67-7c4f-4831-ba7d-b84bef38683c   ○ cadvisor 바이너리 파일 위치 $ cd /var/vcap/packages/cadvisor/   ○ cadvisor 프로세스 확인 $ sudo ps -ef | grep cadvisor   ○ cadvisor 상태 체크 API $ curl 127.0.0.1:8080/healthz   ○ cadvisor 컨테이너 조회 API $ curl 127.0.0.1:8080/api/v1.0/containers   ○ cadvisor 컨테이너 세부 조회 API (실제 garden안에 특정 container 정보 조회 내용) $ curl http://127.0.0.1:8080/api/v1.0/containers/garden/12cb1c9b-96e8-4e0e-5177-7b98   ○ Influxdb 컨테이너 모니터링 정보 $ bosh -e micro-bosh vms  $ bosh -e micro-bosh -d paasta-monitoring ssh influxdb/cf50a8bd-1db0-4d88-8014-71a8e13586a1  $ cd /var/vcap/packages/influxdb  $ ./influx -precision rfc3339  > show databases;  > use container\_metric\_db;  > show measurements;  > select \* from container\_metrics order by desc limit 10; ○ influxdb 전체 컨테이너 모니터링 데이터 개수 확인 > SELECT SUM(count) FROM (SELECT \*,count::INTEGER FROM container\_metrics GROUP BY count FILL(1)) ○ Influxdb 컨테이너 모니터링 하루 총 개수 확인 (10월 24일) > SELECT SUM(count) FROM (SELECT \*,count::INTEGER FROM container\_metrics where time >= '2020-10-23 15:00:00' and time < '2020-10-24 15:00:00' GROUP BY count FILL(1)) |

### **5. cAdvisor 수정사항**

#### 1) 수정사항

|  |
| --- |
| ○ cadvisor 버전 변경  - release-v0.37 버전 적용  ○ cadvisor > cmd > internal > storage > influxdb > influxdb.go  - 컨테이너 필터링 기능  - CF rep API 요청시 인증서 적용  - Influxdb UDP port 적용 (8069)  ○ cadvisor > cmd > internal > storage > influxdb\_origin  - 기존 기능 백업  - influx 모니터링 데이터 적재 시, 각 항목 별 테이블 생성 및 데이터 적재. |

### **6. cAdvisor 실 배포 절차**

#### 1) 과정

|  |
| --- |
| ○ <https://paas-ta.kr/download/package> PaaSTA-Release 클릭  ○ <http://45.248.73.44/index.php/s/wN8KGpqBN8LEFAQ> paasta-monitoring 다운로드  ○ diego-cell 압축해제 및 바이너리 파일 변경  - 1. paasta-monitoring-agent.tgz 파일  - 2. package > cadvisor.tgz파일 압축해제  -3. chmod 755  - 4. cadvisor tar 압축  hso@hso:~/Music/cadvisor\_package$tar czvf ../paasta-monitoring-agent/packages/cadvisor.tgz .  - 5. paasta-monitoring-agent 압축  hso@hso:~/Music/paasta-monitoring-agent$ tar czvf ../latest/paasta-monitoring/paasta-monitoring-agent.tgz jobs/ packages/ Readme.md  - 6. paasta-monitoring 압축  hso@hso:~/Music/latest/paasta-monitoring$ zip ../../paasta-monitoring.zip \*  ○ diego-cell 압축  ○ inception-vm에 해당 paasta deployment 재실행  ○ yml에 환경변수 추가  - --allow\_dynamic\_housekeeping=true --housekeeping\_interval=30s |

### **7. cAdvisor 히스토리**

#### 1) 테스트

|  |
| --- |
| ○ cadvisor build  1. git clone -b v5.1.0-dev <https://github.com/PaaS-TA/PaaS-TA-Monitoring.git>  2. mkdir go  3. export GOPATH=/home/hso/workspace3/go  (위 명령어는 프로젝트 별로 설정해야함)  (아래 명령어는 최초 설정 , go version 1.14.9)  (export GOROOT=/usr/local/go)  (export PATH=$PATH:/usr/local/go/bin)  4. source ~/.bashrc |

|  |
| --- |
| ○ cadvisor build (local)  cadvisor build guide (<https://github.com/google/cadvisor/blob/master/docs/development/build.md>)  0. cd cadvisor  1. make build  2. make test  3. sudo ./cadvisor  ( sudo ./cadvisor -storage\_driver\_host=172.168.1.12:8086 -storage\_driver=influxdb -storage\_driver\_db=cadvisor -storage\_driver\_user='test' -storage\_driver\_password='test' -storage\_driver\_secure=False) |

|  |
| --- |
| ○ cadvisor build (docker container)  0. cd cadvisor  1. docker build -t google/cadvisor:v0.37.1 --file deploy/Dockerfile ../cadvisor  2. sudo docker run --volume=/:/rootfs:ro --volume=/var/run:/var/run:rw --volume=/sys:/sys:ro --volume=/var/lib/docker/:/var/lib/docker:ro --publish=8080:8080 --detach=true --name=cadvisor\_v0.37.1 google/cadvisor:v0.37.1 -storage\_driver\_host=172.168.1.12:8086 -storage\_driver=influxdb -storage\_driver\_db=cadvisor -storage\_driver\_user='test' -storage\_driver\_password='test' -storage\_driver\_secure=False |

|  |
| --- |
| ○ cadvisor build (동작확인)  0. curl <http://172.30.1.6:8080/containers/>  1. curl <http://localhost:8080/healthz>  2. curl <http://127.0.0.1:8080/api/v1.0/containers>  3. curl <http://127.0.0.1:8080/api/v1.0/containers/garden/12cb1c9b-96e8-4e0e-5177-7b98> |

|  |
| --- |
| ○ cadvisor - rep 간 API 요청 에러  Failed get : %s Get “<http://127.0.0.1:1800/v1/containers>” : dial tcp 127.0.0.1:1800: connect: connection refused |

|  |
| --- |
| ○ inception-vm 과 diego-cell vm 간 파일 전송  $ bosh -e micro-bosh -d paasta scp -r diego-cell/45088599-909a-4140-a330-1008b9c72f06:/tmp/ ~/ca/  $ bosh -e micro-bosh -d paasta scp -r cadvisor diego-cell/b4d38bf3-e177-4b6f-afbe-af71ed12543a:/tmp/ |

#### 2) 기존 customizing\_cadvisor 소스 내용 (반영)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ==============  How to test :  ==============  $ cd cadvisor\_home/  $ source .envrc  $ cd src/github.com/google/cadvisor/  $ go install  $ cadvisor --storage\_driver=influxdb --storage\_driver\_db=container\_metric\_db --storage\_driver\_host=192.168.4.16:8069 --storage\_driver\_user=cf\_metric\_admin –storage\_driver\_password=metric\_password **container\_housekeeping\_interval=15s**  ### jobs - cadviosr 추가 ###  # monit   |  | | --- | | check process cadvisor  with pidfile /var/vcap/sys/run/cadvisor/cadvisor.pid  start program "/var/vcap/jobs/cadvisor/bin/cadvisor\_ctl start"  stop program "/var/vcap/jobs/cadvisor/bin/cadvisor\_ctl stop"  group vcap |   # spec   |  | | --- | | ---  name: cadvisor  templates:  cadvisor\_ctl.erb: bin/cadvisor\_ctl  packages:  - pid\_utils  - cadvisor  properties:  cadvisor.storage\_driver:  description: "Storage Driver - ex: influxdb, elasticsearch, kafka, redis etc"  default: "influxdb"  cadvisor.storage\_driver\_db:  description: "Storage Database Name"  default: "cf\_metric\_db"  cadvisor.storage\_driver\_ip:  description: "Storage Server Connection IP"  default: "127.0.0.1"  cadvisor.storage\_driver\_port:  description: "Storage Server connection port"  default: 8089  cadvisor.container\_housekeeping\_interval:  description: "interval to allow between container housekeepings. format : 15s, 15m, 15h. - must attach unit of time."  default: 15s |   # template/cadvisor\_ctl.erb   |  | | --- | | #!/bin/bash -e  RUN\_DIR=/var/vcap/sys/run/cadvisor  LOG\_DIR=/var/vcap/sys/log/cadvisor  PIDFILE=$RUN\_DIR/cadvisor.pid  DATA\_DIR=/var/vcap/data/cadvisor  TMP\_DIR=$DATA\_DIR/tmp  source /var/vcap/packages/pid\_utils/pid\_utils.sh  case $1 in  start)  pid\_guard $PIDFILE "cadvisor"  mkdir -p $RUN\_DIR  chown -R vcap:vcap $RUN\_DIR  mkdir -p $LOG\_DIR  chown -R vcap:vcap $LOG\_DIR  mkdir -p $DATA\_DIR  chown -R vcap:vcap $DATA\_DIR  mkdir -p $TMP\_DIR  chown -R vcap:vcap $TMP\_DIR  export GOMAXPROCS=$(nproc)  #echo $$ > $PIDFILE  PID=$$  echo "current pid : $PID"  echo $PID > $PIDFILE    # change file mode to 0666  chmod 666 $PIDFILE  # Allowed number of open file descriptors  ulimit -n 100000  exec chpst -u vcap:vcap /var/vcap/packages/cadvisor/cadvisor \  -container\_housekeeping\_interval=<%= p("cadvisor.container\_housekeeping\_interval") %> \  -storage\_driver=<%= p("cadvisor.storage\_driver") %> \  -storage\_driver\_db=<%= p("cadvisor.storage\_driver\_db") %> \  -storage\_driver\_host=<%= p("cadvisor.storage\_driver\_ip") %>:<%= p("cadvisor.storage\_driver\_port") %> \  2> >(tee -a $LOG\_DIR/cadvisor.stderr.log | logger -p user.error -t vcap.cadvisor) \  1> >(tee -a $LOG\_DIR/cadvisor.stdout.log | logger -p user.info -t vcap.cadvisor)  ;;  stop)  kill\_and\_wait $PIDFILE  ;;  \*)  echo "Usage: cadvisor\_ctl {start|stop}"  ;;  esac |   ### packages - cadvisor 추가 ###  # packaging   |  | | --- | | set -e -x  cp -a cadvisor/cadvisor ${BOSH\_INSTALL\_TARGET} |   # spec   |  | | --- | | ---  name: cadvisor  files:  - cadvisor/cadvisor |   ### manifest file : ex) diego.yml ###  #cell instance에서 template 추가   |  | | --- | | - instances: 1  name: cell\_z1  ...  templates:  ...  - name: cadvisor  release: diego  ...  update:  max\_in\_flight: 1  serial: false |   #properties 정보에서 cadvisor 관련 추가   |  | | --- | | ##### CAdvisor properties #####  cadvisor:  storage\_driver: influxdb  storage\_driver\_db: container\_metric\_db  storage\_driver\_ip: 52.207.82.208  storage\_driver\_port: 8069  container\_housekeeping\_interval: 15s  ########################################### |   **### Customizing Factor [golang source]###**  # storagedriver 등록  - storage 디렉토리에 존재하는 elasticsearch, influxdb, kafka 등의 package 각각에 init() 함수에서 driver 등록됨.  ex) influxdb.go  **--------------------------------------------------------**  **func** init() {  storage.RegisterStorageDriver(**"influxdb"**, new)  } **--------------------------------------------------------**  # 추가로 필요한 parameter 정보 등록 (common\_flags.go)  **--------------------------------------------------------**  \* 아래 사항 추가  //=====================================  **var** ArgDbHost = flag.String(**"storage\_driver\_host"**, **"localhost:8086"**, **"database host:port"**)  **var** ArgDbName = flag.String(**"storage\_driver\_db"**, **"cadvisor"**, **"database name"**)  **var** ArgDbTable = flag.String(**"storage\_driver\_table"**, **"stats"**, **"table name"**) //======================================= **--------------------------------------------------------**  # container housekeeping interval 추가 (cadvisor.go)  **--------------------------------------------------------** *// 2016.12.19 newly added paramter.* **var** containerHousekeepingInterval = flag.Duration(**"container\_housekeeping\_interval"**, 15\*time.***Second***, **"interval to allow between container housekeepings"**) **--------------------------------------------------------**  # github.com/google/cadvisor/manager/manager.go   |  | | --- | | **type manager struct {**  **...  containerHousekeepingInterval time.Duration //추가**  **... }**  **- func** New 함수에 containerHousekeepingInterval time.Duration 추가  ... newManager := &manager{  containers: make(**map**[namespacedContainerName]\*containerData),  quitChannels: make([]**chan** error, 0, 2),  memoryCache: memoryCache,  fsInfo: fsInfo,  cadvisorContainer: selfContainer,  inHostNamespace: inHostNamespace,  startupTime: time.Now(),  maxHousekeepingInterval: maxHousekeepingInterval,  allowDynamicHousekeeping: allowDynamicHousekeeping,  ignoreMetrics: ignoreMetricsSet,  containerWatchers: []watcher.ContainerWatcher{},  eventsChannel: eventsChannel,  collectorHttpClient: collectorHttpClient,  containerHousekeepingInterval: containerHousekeepingInterval, *//Added parameter - interval for collecting container metrics.*  }  ...  - **func** (m \*manager) createContainerLocked(containerName string, watchSource watcher.ContainerWatchSource) error {  ... // m.containerHousekeepingInterval 추가 cont, err := newContainerData(containerName, m.memoryCache, handler, logUsage, collectorManager, m.maxHousekeepingInterval, m.allowDynamicHousekeeping, m.containerHousekeepingInterval) ...  } |   # github.com/google/cadvisor/manager/container.go   |  | | --- | | newContainerData func에 containerHouseKeepingInterval time.Duration 추가  cont := &containerData{  ...  *//housekeepingInterval: \*HousekeepingInterval,*  housekeepingInterval: containerHouseKeepingInterval, *//customizing housekeepinginterval - it can set yaml . paramter : container\_housekeeping\_interval ...*  }  **func** (c \*containerData) housekeeping() {  *//fmt.Println("##### manager/container.go - containerData- housekeeping() called #####")*  *// Start any background goroutines - must be cleaned up in c.handler.Cleanup().*  c.handler.Start()  **defer** c.handler.Cleanup()  *// Initialize cpuload reader - must be cleaned up in c.loadReader.Stop()*  **if** c.loadReader != nil {  err := c.loadReader.Start()  **if** err != nil {  glog.Warningf(**"Could not start cpu load stat collector for %q: %s"**, c.info.Name, err)  }  **defer** c.loadReader.Stop()  }  *// Long housekeeping is either 100ms or half of the housekeeping interval.*  longHousekeeping := 100 \* time.***Millisecond***  **if** \*HousekeepingInterval/2 < longHousekeeping {  longHousekeeping = \*HousekeepingInterval / 2  }  *// Housekeep every second.*  glog.V(3).Infof(**"Start housekeeping for container %q\n"**, c.info.Name)  *//####### Customizing - annotation below : replace lastHousekeeping to c.housekeepingInterval*  *//lastHousekeeping := time.Now()*  **for** {  **select** {  **case** <-c.stop:  *// Stop housekeeping when signaled.*  **return**  **default**:  *// Perform housekeeping.*  start := time.Now()  *//fmt.Println("============= start time:", start.Unix())*  c.housekeepingTick()  *// Log if housekeeping took too long.*  duration := time.Since(start)  **if** duration >= longHousekeeping {  glog.V(3).Infof(**"[%s] Housekeeping took %s"**, c.info.Name, duration)  }  }  *// Log usage if asked to do so.*  **if** c.logUsage {  **const *numSamples*** = 60  **var** empty time.Time  stats, err := c.memoryCache.RecentStats(c.info.Name, empty, empty, ***numSamples***)  **if** err != nil {  **if** c.allowErrorLogging() {  glog.Infof(**"[%s] Failed to get recent stats for logging usage: %v"**, c.info.Name, err)  }  } **else if** len(stats) < ***numSamples*** {  *// Ignore, not enough stats yet.*  } **else** {  usageCpuNs := uint64(0)  **for** i := **range** stats {  **if** i > 0 {  usageCpuNs += (stats[i].Cpu.Usage.Total - stats[i-1].Cpu.Usage.Total)  }  }  usageMemory := stats[***numSamples***-1].Memory.Usage  instantUsageInCores := float64(stats[***numSamples***-1].Cpu.Usage.Total-stats[***numSamples***-2].Cpu.Usage.Total) / float64(stats[***numSamples***-1].Timestamp.Sub(stats[***numSamples***-2].Timestamp).Nanoseconds())  usageInCores := float64(usageCpuNs) / float64(stats[***numSamples***-1].Timestamp.Sub(stats[0].Timestamp).Nanoseconds())  usageInHuman := units.HumanSize(float64(usageMemory))  glog.Infof(**"[%s] %.3f cores (average: %.3f cores), %s of memory"**, c.info.Name, instantUsageInCores, usageInCores, usageInHuman)  }  }  *//fmt.Println("############################ sleeping before :", time.Since(lastHousekeeping))*  *//Customizing - replace interval time*  time.Sleep(c.housekeepingInterval)  */\**  *next := c.nextHousekeeping(lastHousekeeping)*  *// Schedule the next housekeeping. Sleep until that time.*  *if time.Now().Before(next) {*  *time.Sleep(next.Sub(time.Now()))*  *} else {*  *next = time.Now()*  *}*  *lastHousekeeping = next*  *\*/*  *//fmt.Println("############################ sleeping after :", time.Since(lastHousekeeping))*  }  } |   # influxdb.go 수정 (기존 http 방식에서 udp 방식으로 변경 및 rep 연동 추가) - 소스전체   |  | | --- | | *// Copyright 2014 Google Inc. All Rights Reserved.*  *//*  *// Licensed under the Apache License, Version 2.0 (the "License");*  *// you may not use this file except in compliance with the License.*  *// You may obtain a copy of the License at*  *//*  *// http://www.apache.org/licenses/LICENSE-2.0*  *//*  *// Unless required by applicable law or agreed to in writing, software*  *// distributed under the License is distributed on an "AS IS" BASIS,*  *// WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.*  *// See the License for the specific language governing permissions and*  *// limitations under the License.*  **package** influxdb  **import** (  **"os"**  **"sync"**  **"time"**  **"net/http"**  **"io/ioutil"**  **"encoding/json"**  **"errors"**  **"strings"**  **"net"**  **"github.com/golang/glog"**  **"github.com/google/cadvisor/storage"**  info **"github.com/google/cadvisor/info/v1"**  influxdb **"github.com/influxdata/influxdb/client/v2"**  )  **func** init() {  storage.RegisterStorageDriver(**"influxdb"**, new)  }  **type** influxdbStorage **struct** {  client influxdb.Client  cellIp string  machineName string  database string  retentionPolicy string  bufferDuration time.Duration  lastWrite time.Time  points []\*influxdb.Point  lock sync.Mutex  readyToFlush **func**() bool  }  *//====================================================================================*  *// Container Metrics Metadata from REP (127.0.0.1:1800/v1/containers)*  **type** ContainerMetricsMetadata **struct**{  Limits Limits **`json:"limits,omitempty"`**  UsageMetrics UsageMetrics **`json:"usage\_metrics,omitempty"`**  Container\_Id string **`json:"container\_id,omitempty"`**  Application\_Id string **`json:"application\_id,omitempty"`**  Application\_Index string **`json:"application\_index,omitempty"`**  Application\_Name string **`json:"application\_name,omitempty"`**  Application\_Urls []string **`json:"application\_uris,omitempty"`**  }  **type** Limits **struct** {  Fds int32 **`json:"fds,omitempty"`**  Memory int32 **`json:"mem,omitempty"`**  Disk int32 **`json:"disk,omitempty"`**  }  **type** UsageMetrics **struct** {  MemoryUsageInBytes uint64 **`json:"memory\_usage\_in\_bytes"`**  DiskUsageInBytes uint64 **`json:"disk\_usage\_in\_bytes"`**  TimeSpentInCPU time.Duration **`json:"time\_spent\_in\_cpu"`**  }  *//====================================================================================*  *// Series names*  **const** (  *// Cumulative CPU usage*  ***serCpuUsageTotal*** string = **"cpu\_usage\_total"**  ***serCpuUsageSystem*** string = **"cpu\_usage\_system"**  ***serCpuUsageUser*** string = **"cpu\_usage\_user"**  ***serCpuUsagePerCpu*** string = **"cpu\_usage\_per\_cpu"**  *// Smoothed average of number of runnable threads x 1000.*  ***serLoadAverage*** string = **"load\_average"**  *// Memory Usage*  ***serMemoryUsage*** string = **"memory\_usage"**  *// Working set size*  ***serMemoryWorkingSet*** string = **"memory\_working\_set"**  *// Cumulative count of bytes received.*  ***serRxBytes*** string = **"rx\_bytes"**  *// Cumulative count of receive errors encountered.*  ***serRxErrors*** string = **"rx\_errors"**  *// Cumulative count of bytes transmitted.*  ***serTxBytes*** string = **"tx\_bytes"**  *// Cumulative count of transmit errors encountered.*  ***serTxErrors*** string = **"tx\_errors"**  ***serRxDropped*** string = **"rx\_dropped"**  ***serTxDropped*** string = **"tx\_dropped"**  *// Filesystem device.*  *//serFsDevice string = "fs\_device"*  *// Filesystem limit.*  *//serFsLimit string = "fs\_limit"*  *// Filesystem usage.*  *//serFsUsage string = "fs\_usage"*  *// Disk Usage*  ***serDiskUsage*** string = **"disk\_usage"**  *// Container Measurement*  ***serContainerMeausement*** string = **"container\_metrics"**  )  **func** new() (storage.StorageDriver, error) {  hostname, err := os.Hostname()  **if** err != nil {  **return** nil, err  }  **var** cellIp string  addrs, err := net.InterfaceAddrs()  **if** err != nil {  **return** nil, err  }  **for** \_, address := **range** addrs {  *// check the address type and if it is not a loopback the display it*  **if** ipnet, ok := address.(\*net.IPNet); ok && !ipnet.IP.IsLoopback() {  **if** ipnet.IP.To4() != nil {  cellIp = ipnet.IP.String() *//fmt.Println(ipnet.IP.String())*  }  }  }  **return** newStorage(  hostname,  cellIp,  \*storage.ArgDbTable,  \*storage.ArgDbName,  \*storage.ArgDbHost,  )  }  *// machineName: A unique identifier to identify the host that current cAdvisor*  *// instance is running on.*  *// influxdbHost: The host which runs influxdb (host:port)*  **func** newStorage(  machineName,  cellIp,  tablename,  database,  influxdbHost string,  ) (\*influxdbStorage, error) {  *// Make client*  client, err := influxdb.NewUDPClient(influxdb.UDPConfig{  Addr: influxdbHost,  *//PayloadSize: 4096,*  })  **if** err != nil {  **return** nil, err  }  ret := &influxdbStorage{  client: client,  machineName: machineName,  cellIp: cellIp,  database: database,  lastWrite: time.Now(),  points: make([]\*influxdb.Point, 0),  }  ret.readyToFlush = ret.defaultReadyToFlush  **return** ret, nil  }  *// Field names*  **const** (  ***fieldAppDisk*** string = **"app\_disk"**  ***fieldAppMem*** string = **"app\_mem"**  ***fieldValue*** string = **"value"**  ***fieldType*** string = **"type"**  ***fieldDevice*** string = **"device"**  )  *// Tag names*  **const** (  ***tagName*** string = **"name"**  ***tagMachineName*** string = **"machine"**  ***tagContainerName*** string = **"container\_name"**  ***tagCellIp*** string = **"cell\_ip"**  ***tagApplicationId*** string = **"application\_id"**  ***tagApplicationIndex*** string = **"application\_index"**  ***tagApplicationName*** string = **"application\_name"**  ***tagApplicationUrl*** string = **"application\_url"**  )  *//====================================================================================*  *// Container Metrics Metadata from REP (127.0.0.1:1800/v1/containers)*  **func** (self \*influxdbStorage) containerMetricsMedataData() []ContainerMetricsMetadata{  client := &http.Client{  CheckRedirect: **func**(req \*http.Request, \_ []\*http.Request) error {  *//dumpRequest(req)*  **return** errors.New(**"No redirects"**)  },  Timeout: 30 \* time.***Second***,  Transport: &http.Transport{  DisableKeepAlives: true,  TLSHandshakeTimeout: 10 \* time.***Second***,  },  }  reqUrl := **"http://127.0.0.1:1800/v1/containers"**  req, err := http.NewRequest(**"GET"**, reqUrl, nil)  resp, err := client.Do(req)  **if** err != nil {  glog.Error(**"##### get Container Metrics Metadata request err:"**, err)  }  **if** resp != nil{  rawdata, \_ := ioutil.ReadAll(resp.Body)  *//fmt.Println("##### Response Data :", string(rawdata))*  **var** containermetrics []ContainerMetricsMetadata  json.Unmarshal(rawdata, &containermetrics)  */\*fmt.Println("##### Container Metrics Metadata :", containermetrics, len(containermetrics))*  *for \_, metrics :=range containermetrics{*  *fmt.Println("##### Container Metrics container id :", metrics.Container\_Id)*  *fmt.Println("##### Container Metrics app id :", metrics.Application\_Id)*  *fmt.Println("##### Container Metrics app name :", metrics.Application\_Name)*  *fmt.Println("##### Container Metrics app urls :", metrics.Application\_Urls)*  *fmt.Println("##### Container Metrics app limits :", metrics.Limits)*  *fmt.Println("##### Container Metrics app usage-memory :", metrics.UsageMetrics.MemoryUsageInBytes)*  *fmt.Println("##### Container Metrics app usage-disk :", metrics.UsageMetrics.DiskUsageInBytes)*  *fmt.Println("##### Container Metrics app usage-cpu(second) :", metrics.UsageMetrics.TimeSpentInCPU.Seconds())*  *}\*/*  **return** containermetrics  }  **return** nil  }  *//====================================================================================*  **func** (self \*influxdbStorage) containerFilesystemStatsToPoints(  *//ref info.ContainerReference,*  containerName string,  stats \*info.ContainerStats) (points []\*influxdb.Point) {  **if** len(stats.Filesystem) == 0 {  **return** points  }  **for** \_, fsStat := **range** stats.Filesystem {  tagsFsUsage := **map**[string]string{  ***tagMachineName***: self.machineName,  ***tagContainerName***: containerName,  ***fieldDevice***: fsStat.Device,  ***fieldType***: **"usage"**,  }  fieldsFsUsage := **map**[string]**interface**{}{  ***fieldValue***: float64(fsStat.Usage),  }  fsUsagePt, err :=influxdb.NewPoint(***serContainerMeausement***, tagsFsUsage, fieldsFsUsage)  **if** err != nil {  glog.Fatalf(**"Failed to create NewPoint for FieldsFsUsage: %v"**, err)  }  tagsFsLimit := **map**[string]string{  ***tagMachineName***: self.machineName,  ***tagContainerName***: containerName,  ***fieldDevice***: fsStat.Device,  ***fieldType***: **"limit"**,  }  fieldsFsLimit := **map**[string]**interface**{}{  ***fieldValue***: float64(fsStat.Limit),  }  fsLimitPt, err := influxdb.NewPoint(***serContainerMeausement***, tagsFsLimit, fieldsFsLimit)  **if** err != nil {  glog.Fatalf(**"Failed to create NewPoint for FieldsFsLimit: %v"**, err)  }  points = append(points, fsUsagePt, fsLimitPt)  }  *//self.tagPoints(ref, stats, points)*  **return** points  }  **func** (self \*influxdbStorage) containerStatsToPoints(  *//ref info.ContainerReference,*  containerName string,  containerMetric ContainerMetricsMetadata,  stats \*info.ContainerStats,  ) (points []\*influxdb.Point) {  */\**  *//============================= rep (/v1/containers)로부터* *container resource usage metrics 정보를 전달받아* *influxdb에 저장한다. ============================*  *// rep로부터 전달받는* *mertrics : cpu, memory, disk usage*  *// CPU usage: Total usage in nanoseconds*  *points = append(points, makePoint(self.machineName, self.cellIp, containerName, serCpuUsageTotal, containerMetric, float64(stats.Cpu.Usage.Total)))*  *// CPU usage: Time spend in system space (in nanoseconds)*  *points = append(points, makePoint(self.machineName, self.cellIp, containerName, serCpuUsageSystem, containerMetric, float64(stats.Cpu.Usage.System)))*  *// CPU usage: Time spent in user space (in nanoseconds)*  *points = append(points, makePoint(self.machineName, self.cellIp, containerName, serCpuUsageUser, containerMetric, float64(stats.Cpu.Usage.User)))*  *// CPU usage per CPU*  *for i := 0; i < len(stats.Cpu.Usage.PerCpu); i++ {*  *point := makePoint(self.machineName, self.cellIp, containerName, serCpuUsagePerCpu, containerMetric, float64(stats.Cpu.Usage.PerCpu[i]))*  *\*//\*tags := map[string]string{"instance": fmt.Sprintf("%v", i)}*  *addTagsToPoint(point, tags)\*//\**  *points = append(points, point)*  *}*  *// Load Average*  *points = append(points, makePoint(self.machineName, self.cellIp, containerName, serLoadAverage, containerMetric, float64(stats.Cpu.LoadAverage)))*  *// Memory Usage*  *points = append(points, makePoint(self.machineName, self.cellIp, containerName, serMemoryUsage, containerMetric, float64(stats.Memory.Usage)))*  *// Working Set Size*  *points = append(points, makePoint(self.machineName, self.cellIp, containerName, serMemoryWorkingSet, containerMetric, float64(stats.Memory.WorkingSet)))*  *\*/*  *// CPU Usage*  points = append(points, makePoint(self.machineName, self.cellIp, containerName, ***serCpuUsageTotal***, containerMetric, containerMetric.UsageMetrics.TimeSpentInCPU.Seconds()))  *// Load Average*  points = append(points, makePoint(self.machineName, self.cellIp, containerName, ***serLoadAverage***, containerMetric, float64(stats.Cpu.LoadAverage)))  *// Memory Usage*  points = append(points, makePoint(self.machineName, self.cellIp, containerName, ***serMemoryUsage***, containerMetric, float64(containerMetric.UsageMetrics.MemoryUsageInBytes)))  *// Disk Usage*  points = append(points, makePoint(self.machineName, self.cellIp, containerName, ***serDiskUsage***, containerMetric, float64(containerMetric.UsageMetrics.DiskUsageInBytes)))  *// Network Stats*  **for** i := 0 ; i < len(stats.Network.Interfaces); i ++ {  */\*fmt.Println("interface name :", stats.Network.Interfaces[i].Name)*  *fmt.Println("rxbytes :", stats.Network.Interfaces[i].RxBytes)*  *fmt.Println("rxerror :", stats.Network.Interfaces[i].RxErrors)*  *fmt.Println("rxdropped :", stats.Network.Interfaces[i].RxDropped)*  *fmt.Println("txbytes :", stats.Network.Interfaces[i].TxBytes)*  *fmt.Println("txerror :", stats.Network.Interfaces[i].TxErrors)*  *fmt.Println("txdropped :", stats.Network.Interfaces[i].TxDropped)\*/*  points = append(points, makePoint(self.machineName, self.cellIp, stats.Network.Interfaces[i].Name, ***serRxBytes***, containerMetric, float64(stats.Network.Interfaces[i].RxBytes)))  points = append(points, makePoint(self.machineName, self.cellIp, stats.Network.Interfaces[i].Name, ***serRxErrors***, containerMetric, float64(stats.Network.Interfaces[i].RxErrors)))  points = append(points, makePoint(self.machineName, self.cellIp, stats.Network.Interfaces[i].Name, ***serRxDropped***, containerMetric, float64(stats.Network.Interfaces[i].RxDropped)))  points = append(points, makePoint(self.machineName, self.cellIp, stats.Network.Interfaces[i].Name, ***serTxBytes***, containerMetric, float64(stats.Network.Interfaces[i].TxBytes)))  points = append(points, makePoint(self.machineName, self.cellIp, stats.Network.Interfaces[i].Name, ***serTxErrors***, containerMetric, float64(stats.Network.Interfaces[i].TxErrors)))  points = append(points, makePoint(self.machineName, self.cellIp, stats.Network.Interfaces[i].Name, ***serTxDropped***, containerMetric, float64(stats.Network.Interfaces[i].TxDropped)))  }  *//self.tagPoints(ref, stats, points)*  **return** points  }  **func** (self \*influxdbStorage) OverrideReadyToFlush(readyToFlush **func**() bool) {  self.readyToFlush = readyToFlush  }  **func** (self \*influxdbStorage) defaultReadyToFlush() bool {  **return** time.Since(self.lastWrite) >= self.bufferDuration  }  **func** (self \*influxdbStorage) AddStats(ref info.ContainerReference, stats \*info.ContainerStats) error {  *//fmt.Println("##### influxdb.go - AddStats called #####")*  **if** stats == nil {  **return** nil  }  **var** pointsToFlush []\*influxdb.Point  **func**() {  *// AddStats will be invoked simultaneously from multiple threads and only one of them will perform a write.*  self.lock.Lock()  **defer** self.lock.Unlock()  **var** containerName string  **var** containerMetric ContainerMetricsMetadata  **if** len(ref.Aliases) > 0 {  containerName = ref.Aliases[0]  } **else** {  containerName = ref.Name  }  *//===================================================================*  *// Container Metrics Metadata from REP (127.0.0.1:1800/v1/containers)*  containerMetrics := self.containerMetricsMedataData()  containerNames := strings.Split(containerName, **"-"**)  containerMetric.Container\_Id = containerNames[len(containerNames) -1]  **for** \_, metrics := **range** containerMetrics{  **if** metrics.Container\_Id == containerMetric.Container\_Id {  containerMetric.Application\_Id = metrics.Application\_Id  containerMetric.Application\_Name = metrics.Application\_Name  containerMetric.Application\_Urls = metrics.Application\_Urls  containerMetric.Application\_Index = metrics.Application\_Index  containerMetric.Limits.Disk = metrics.Limits.Disk  containerMetric.Limits.Memory = metrics.Limits.Memory  containerMetric.UsageMetrics.MemoryUsageInBytes = metrics.UsageMetrics.MemoryUsageInBytes  containerMetric.UsageMetrics.DiskUsageInBytes = metrics.UsageMetrics.DiskUsageInBytes  containerMetric.UsageMetrics.TimeSpentInCPU = metrics.UsageMetrics.TimeSpentInCPU  }  }  *//===================================================================*  self.points = append(self.points, self.containerStatsToPoints(containerName, containerMetric, stats)...)  self.points = append(self.points, self.containerFilesystemStatsToPoints(containerName, stats)...)  **if** self.readyToFlush() {  pointsToFlush = self.points  self.points = make([]\*influxdb.Point, 0)  self.lastWrite = time.Now()  }  }()  **if** len(pointsToFlush) > 0 {  *// Create a new point batch*  bp, err := influxdb.NewBatchPoints(influxdb.BatchPointsConfig{  Database: self.database,  Precision: **"s"**,  })  **if** err != nil {  glog.Fatalf(**"Failed to create NewBatchPoint: %v"**, err)  }  *//points := make([]influxdb.Point, len(pointsToFlush))*  **for** \_, p := **range** pointsToFlush {  *//points[i] = \*p*  *//fmt.Println("point to save at database ",self.database, i, p)*  bp.AddPoint(p)  }  err = self.client.Write(bp)  **if** err != nil {  glog.Fatalf(**"Failed to send point to influxdb: %v"**, err)  }  }  **return** nil  }  **func** (self \*influxdbStorage) Close() error {  self.client = nil  **return** nil  }  *// Creates a measurement point with a single value field*  **func** makePoint(machineName, cellIp, containerName, name string, containerMetric ContainerMetricsMetadata, value float64) \*influxdb.Point {  **var** tags **map**[string]string  **var** fields **map**[string]**interface**{}  **if** containerMetric.Application\_Id != **""** {  tags = **map**[string]string{  ***tagName***: name,  ***tagMachineName***: machineName,  ***tagCellIp***: cellIp,  ***tagContainerName*** : containerName,  ***tagApplicationId*** : containerMetric.Application\_Id,  ***tagApplicationIndex***: containerMetric.Application\_Index,  ***tagApplicationName*** : containerMetric.Application\_Name,  ***tagApplicationUrl*** : containerMetric.Application\_Urls[0],  }  }**else**{  tags = **map**[string]string{  ***tagName***: name,  ***tagMachineName***: machineName,  ***tagCellIp***: cellIp,  ***tagContainerName*** : containerName,  }  }  **if** containerMetric.Application\_Id != **""** {  fields = **map**[string]**interface**{}{  ***fieldValue***: value,  ***fieldAppDisk*** : float64(containerMetric.Limits.Disk\*1024\*1024),  ***fieldAppMem*** : float64(containerMetric.Limits.Memory\*1024\*1024),  }  }**else**{  fields = **map**[string]**interface**{}{  ***fieldValue***: value,  }  }  mkPoint, err := influxdb.NewPoint(***serContainerMeausement***, tags, fields)  **if** err != nil {  glog.Fatalf(**"Failed to create NewPoint for FieldsFsLimit: %v"**, err)  }  **return** mkPoint  }  *// Some stats have type unsigned integer, but the InfluxDB client accepts only signed integers.*  **func** toSignedIfUnsigned(value **interface**{}) **interface**{} {  **switch** v := value.(**type**) {  **case** uint64:  **return** int64(v)  **case** uint32:  **return** int32(v)  **case** uint16:  **return** int16(v)  **case** uint8:  **return** int8(v)  **case** uint:  **return** int(v)  }  **return** value  } | |

1. 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.) [↑](#footnote-ref-1)