|  |
| --- |
| **클라우드컴퓨팅** |
| **과제 #01 문제 및 보고서** |

|  |  |
| --- | --- |
| **이름** | 이풍헌 |
| **학번** | 201924548 |
| **소속**  **학과/대학** | 정보컴퓨터공학과 |
| **분반** | 059 |

## <주의사항>

* 개별 과제 입니다. (팀으로 진행하는 과제가 아니며, 모든 학생이 보고서를 제출해야 함)
* 파일명에 본인의 이름과 학번을 입력하세요.
* 각각의 문제 바로 아래에 답을 작성 후 제출해 주세요.
  + 소스코드/스크립트 등을 작성 한 경우, 본 문서에 코드를 붙여 넣거나 또는 별도의 파일로 첨부해서 제출하세요.
* PLATO 제출
  + 데드라인을 지나서 제출하면 0점 (데드라인은 PLATO에서 확인하세요)
  + 부정행위 적발 시, 원본(보여준 사람)과 복사본(베낀 사람) 모두 0점 처리함
* PLATO에 아래의 파일을 제출 해 주세요
  + 보고서(PDF 파일로 변환 후 제출을 권장하나 워드 문서 제출도 가능하며, 그 외 형식은 제출 불가)
    - PDF, DOC, DOCX 이외의 형식으로 제출한 경우, 윈도우PC를 사용하는 채점자의 컴퓨터에서 파일이 열리지 않으면 0점으로 처리함
  + 보고서 파일명에 이름과 학번을 입력해 주세요.
  + 소스코드, 스크립트, Makefile 등을 작성해야 하는 경우, 작성한 모든 파일을 본 문서에 붙여 넣기 하거나 또는 하나의 첨부파일(zip 압축파일)로 제출
    - 소스코드를 첨부하도록 안내가 된 경우, 소스코드 미제출 시 20% 감점

## <실습 과제: 기본>

|  |
| --- |
| **[Q 1] 가상 머신 생성 [20 점]**  VirtualBox에서 가상 머신을 생성하고 우분투를 설치하세요(우분투 리눅스는 권장 사항이며, 다른 Linux Distro를 설치해도 됩니다). 설치가 완료되면, 가상 머신을 시작하고 우분투에 로그인 한 후 터미널에서 아래의 명령어를 입력하세요. 명령어 실행 후, 터미널 출력 화면을 캡처해서 본 문서에 첨부하세요. 그리고, 각 명령어는 무슨 동작을 하는 것인지, 출력 결과가 의미하는 것이 무엇인지 설명하세요.  1) lsb\_release –a  2) cat /proc/cpuinfo | grep processor | tail -n 1  3) cat /proc/meminfo | head -n 1 |
| **답변 예시) ls -al > msg.txt; cat msg.txt**  **(화면 캡처)**    **(명령어 설명)** 파일/폴더 목록을 조회하는데(ls) 상세 정보까지 조회하고(-al) 그 결과를 msg.txt라는 파일에 기록함(> msg.txt). 그 다음으로, msg.txt 파일에 저장된 내용을 터미널에 출력함(cat msg.txt)  **(결과 설명)** 터미널 출력 결과에 따르면, 현재 폴더에는 총 4개의 파일이 저장되어 있음 |

답변 1) (터미널 화면 캡처 + 명령어 및 결과 설명): 텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Lsb\_release -a 를 실행하면 Linux standard base 정보를 표시한다.

Linux의 배포판 버전, 코드명 등을 확인할 수 있다.

Linux kali의 버전이 설명되어있다.

답변 2) (터미널 화면 캡처 + 명령어 및 결과 설명):

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

Cat /proc/cpuinfo : cpu정보가 저장된 파일을 보여준다

Grep processor : processor가 들어간 문장을 찾는다

Tail -n 1 : 마지막 줄만 가져온다.

이 가상머신은 1개의 processor만 가지고 있다.

답변 3) (터미널 화면 캡처 + 명령어 및 결과 설명):

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

cat /proc/meminfo : 메모리 정보가 저장된 파일을 표사

head -n 1 : 첫 줄만 가져온다

이 가상머신은 2024164kb 메모리만 사용한다.

|  |
| --- |
| **[Q 2] 가상 머신 ping [20 점]**  호스트 머신에서 가상 머신을 구동하고, 가상 머신의 NIC를 “NAT”로 설정하세요. 리눅스 가상 머신 터미널에서 ifconfig 명령을 입력하고, 터미널 출력 결과를 확인하세요. (ifconfig 명령어가 설치되어 있지 않은 경우, 오류 메시지가 출력됩니다. 이 때는, sudo apt update; sudo apt install net-tools 명령어로 관련 프로그램을 먼저 설치 후, 다시 시도하세요.) 그리고 ifconfig 명령어의 기능을 설명하세요.   1. 호스트 머신의 IP 주소는? 2. 가상 머신의 IP 주소는? 3. 호스트 머신에서 가상 머신으로 PING 을 보내는 명령을 실행하고, 터미널 화면을 캡처하여 아래에 첨부하세요. 호스트 머신에서 가상 머신으로 PING을 보내는 것이 가능한가요? (불가능 한 경우, 그 이유도 설명하시오) 4. 가상 머신에서 호스트 머신으로 PING을 보내는 명령을 실행하고, 터미널 화면을 캡처하여 아래에 첨부하세요. 가상 머신에서 호스트 머신으로 PING을 보내는 것이 가능한가요? (불가능 한 경우, 그 이유도 설명하시오) |

답변 1): 192.168.219.101

답변 2): 10.0.2.15

답변 3):

텍스트, 폰트, 스크린샷, 디자인이(가) 표시된 사진

자동 생성된 설명

Nat로 연결되어 있으면 가상머신은 사설 ip를 받기에 특정불가능하다. 그렇기에 ping이 불가능하다.

답변 4):

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

가능하다.

|  |
| --- |
| **[Q 3] stress 유틸리티 [10 점]**  아래의 명령을 실행하여, 필요한 도구 및 라이브러리를 설치하세요.   * sudo apt update; sudo apt install stress -y * pip install matplotlib * pip install pyutil   아래의 코드를 실행하고 그 결과로 출력되는 그래프를 본 문서에 첨부하세요. 참고로, 아래의 코드에서 N\_CPU 값은 본인의 가상 머신에 할당된 CPU 수로 설정해야 합니다. 프로그램을 실행한 뒤 멈춤 현상이 발생하면 CTRL+C로 프로그램을 중단하고 다시 실행하세요. |
| **import psutil**  **import time**  **import os**  **import matplotlib.pyplot as plt**  **from multiprocessing import Process**  **WARM\_UP\_TIME = 2**  **TIME\_LIMIT = 20**  **WAITING\_TIME = 5**  **STRESS\_PERIOD = 10**  **N\_CPU = 8 # replace this with your CPU count**  **def f\_stress():**  **print("f\_stress started")**  **os.system("stress -c {} -t {}".format(N\_CPU, STRESS\_PERIOD))**  **print("f\_stress finished")**  **cpus=[]**  **for t in range(WARM\_UP\_TIME):**  **print("[warm-up] current time: ", t)**  **time.sleep(1)**  **for t in range(TIME\_LIMIT):**  **print("current time: ", t)**  **if t == WAITING\_TIME:**  **p = Process(target=f\_stress)**  **p.start()**  **curr\_cpu = psutil.cpu\_percent()**  **cpus.append(curr\_cpu)**  **time.sleep(1)**  **p.join()**  **#print(cpus)**  **plt.plot(cpus)**  **plt.xlabel("time (second)")**  **plt.ylabel("CPU usage (percent)")**  **plt.title("CPU usage history")**  **plt.show()** |

답변 (CPU 사용률 변동 그래프 첨부):

텍스트, 도표, 스크린샷, 그래프이(가) 표시된 사진

자동 생성된 설명

Assign1.py

|  |
| --- |
| **[Q 4] Auto-Scaling with VBox [50 점]**  VirtualBox SDK, API, CLI 등을 활용하여 VM Auto-Scaling을 구현하세요.   * 한 개의 VM을 운영중인 상황을 가정하고, 다음의 Auto-Scaling Rule을 사용하여 자동화된 Auto-Scaling을 구현하세요.   + IF( 최근 5초간 CPU 평균 사용률이 70%를 초과 ) THEN 가상 머신 한 개 추가   + 본 문제에서는 수평 확장 만을 고려하며, 축소는 고려하지 않습니다. * CPU 부하 생성을 위해 리눅스 stress 도구를 사용하세요(위 문제에서 사용한 코드를 활용하면 됩니다). * 특정한 서비스를 구동하고, 유효한 서비스 요청을 발생시키는 상황이 아니기 때문에, Auto-Scaling 이후에 Load Balancing, Routing 등은 구현할 필요 없습니다. 확장 조건이 만족되면 자동으로 VM이 생성되어 실행된다는 것을 보이기만 하면 됩니다. * Auto-Scaling 조건 만족 시 VM이 자동으로 생성되는 것을 확인하기 위해, 호스트 PC에서 구동되는 별도의 프로그램 또는 스크립트 파일을 만들고, 1초 간격으로 ACTIVE 가상 머신 목록을 조회하세요. CPU 부하를 고의로 생성하여 Auto-Scaling 조건이 만족되면, 새로운 VM이 생성되고 활성화되어 ACTIVE 가상 머신 수가 증가된다는 것을 터미널 출력을 통해서 보여줘야 합니다. * 본 문제를 위해서 작성한 모든 소스코드를 제출하세요. |

답변 (VM 수가 증가하는 것을 보여주는 터미널 출력을 캡처하여 첨부):

텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명

답변 (Auto-Scaling 규칙을 정의하고 구현한 소스코드의 파일명은 무엇인지 답하시오):

auto.py

답변 (VM 수가 증가하는 것을 보여주기 위해 작성한 소스코드의 파일명은 무엇인지 답하시오):

showVms.py

**끝! 수고하셨습니다 ☺**