IT부문 기타공유

- RDP원격접속시 인증오류
- virtualBox에서 windows11 설치하기
- Fedora23 에 openxenmanager 설치하기
- 리눅스를 데스크탑처럼 사용기.
- 사이트신뢰성엔지니어링(SRE)의 요약
- 크롬브라우져에서 공룡게임하기
- 관측 가능성 (Obserability) 요약

RDP원격접속시 인증오류

RDP를 이용한 상용/개발서버 접근시 첨부된 이미지처럼 "인증오류가 발생했습니다. \n 요청한 함수가 지원되지 않습니다" 라는 팝업창이 뜨는경우 조치방안

1. 원인 : 접속하려는 PC / 윈도우서버의 업데이트에 차이에 의한 접근 실패로 분석 (KB4093120을 포함한 추가 업데이트 포함....)
2. **해결방안 : 접속하려는 업무용 개인PC**
   - 시작메뉴 → 실행 → gpedit.msc 명령어 실행 (로컬그룹 정책 편집기)
   - 좌측 "로컬 컴퓨터 정책" → " 컴퓨터 구성" → "관리템플릿" → "시스템" → "자격증명관리" 까지 진입
   - 우측 설정항목중 **"Oracle 수정 암호화항목"**에 대해 아래내용으로 변경
   - **상태 : 사용 / 보호수준: 취약**

   - 확인 버튼 으로 창 닫기
   - 정책적용 : 시작메뉴 → 실행 → gpupdate /force 실행 (로컬그룹 정책 편집기)

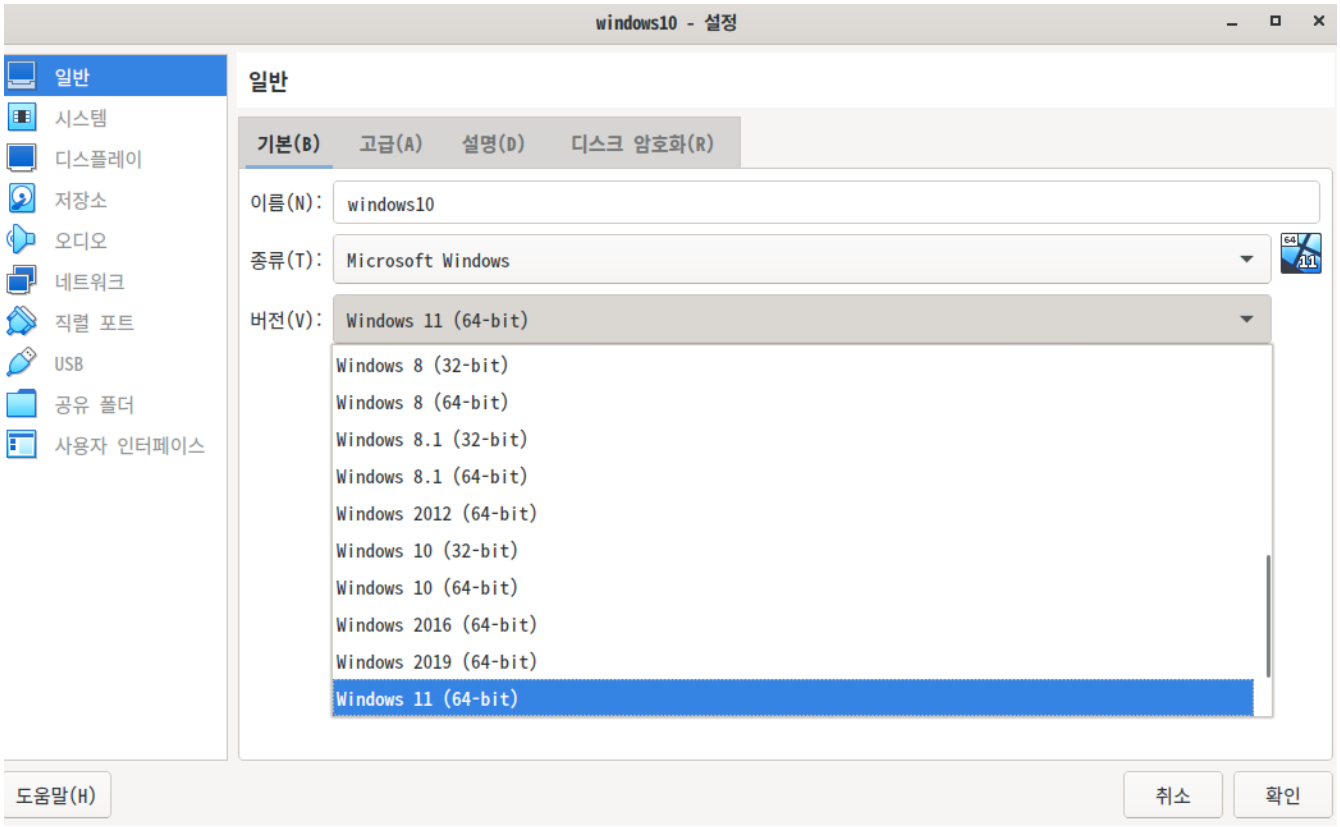
virtualBox에서 windows11 설치하기

VirtualBox 6.1이상에서 windows11 설치시 충족하지 않는나는 메시지 띄우면서 설치가 안되요. (분명 설치 가능한 버전에 Windows 11이 표시되어 있는데.....ㅠㅠ

Windows11은 TPM드라이버가 구성되어 있어야 하는데, VirtualBox에서는 TPM이 정상적으로 인식되지 않아 생기는 문제 같아요

하지만 설치할 수 있는 방법은 있어요

1. VirtualBox에서 Win11으로 구성



2. 메모리는 4G이상, 디스크는 60G이상할당 필요합니다
3. OS설치화면에서 Shift+f10 키 눌러서 커맨드창 나오면 됩니다.
4. 거기에서 regedit 실행
5. 왼쪽 위치는 HKEY_LOCAL_MACHINE\SYSTEM\Setup까지 이동해서 오른쪽 버튼 클릭 ->< 새로운 키 "키 이름은 LabConfig"
6. LabConfig 경로 밑에 문자열을 추가로 만들어야 합니다.
    1. Dword(32bit) : BypassTPMCheck : 1
    2. Dword(32bit) : BypassSecureBootCheck : 1
    3. Dword(32bit) : BypassRamCheck : 1
7. 모든 설정이 끝나면 regedit, cmd 창 닫고 OS설치과정에서 OS설치를 진행하면 됩니다.~

끝.

Reference

https://geekflare.com/windows-11-in-virtual-box/

Fedora23 에 openxenmanager 설치하기

리눅스 환경에서 사용할 수 있는 Xencenter(OpenXenManager) 설치 가이드 입니다.

1. 패키지 설치

```
$> yum install rrdtool pygtk2 gtk-vnc libgtk-vnc- libgvnc python gtk-vnc-python python-configobj
```

2. Openxenmanager내려받기

```
$> git clone https://github.com/OpenXenManager/openxenmanager.git
```

3. 실행하기

```
$> cd openxenmanager
$> python openxenmanager
```

리눅스를 데스크탑처럼 사용기.

노트북에 리눅스를 설치하여 사용한지 어느덧 3년이 되어 갑니다.
처음 구입당시 노트북부터 FreeDos(OS가 없는 녀석)으로 왔기 때문에 윈도우를 사용하려면 따로 구매를 해야했죠.(아! 물론, 윈도우10을 구입했습니다.)

예전에.. 그러니깐 아무 오래전.... 2010년쯤이였을것 같네요. 그때도 노트북에 리눅스를 설치해서 사용해봤는데. 그때는 음... 대충 일주일써보고 안되겠다 싶어서 다시 윈도우로 돌아왔었습니다.ㅋㅋㅋ

그러고 나서 10년뒤... 결론만 이야기 하자면, 무거운 게임을 돌린다거나 하는 일이 아니면 노트북에 리눅스 설치해도 크게 불편함은 느끼지 못할 정보도 엄청나게 발전을 했습니다.

대충.. 제 사양을 적어보면

장비 - HP 파빌리온 노트북, 24인치 모니터 조합
OS : Centos 7.4 -> Rocky Linux 8.2 -> 9.1로 업데이트 하면서 살아오고 있습니다.

저 조합으로 충분하냐? 놉~!! 절대로요..
제가 해결할수 없었던 문제과 해결방안에 대해 좀 적어볼께요.

1. 오피스
    1. 윈도우시절 MS 오피스군(Excel, PowerPotint, Word)를 제일 많이 사용했는데, libreoffice, wps, openoffice 등등등 써보았지만, 호환성문제(워드는 테이블 위치와 크기, 파워포인트는 줄간격 등등등등) 혼자서만 일하는게 아니다보니 막힘이 있었는데 회사에서는 리눅스용 오피스프로그램 제공계획이 없어서 상용Office를 구입하고 나서 해결을 했습니다.
2. 게임
    1. 사람이 어찌 일만 하면서 살아가나요~ 한번씩 머리도 식혀줘야죠..ㅋㅋ
    윈도우때는 스타크래프트1와 스팀에서 판매하는 게임만 가끔 했었는데, RHEL 8버전때는 Win, playonlinux를 사용하면서 한글의 소중함과 아쉬움을 가지면서 살았는데 최근에 9.1로 업그레이드 하고 나서 bottle이라는 녀석을 만나 아주 재미있게 살고 있네요.. bottle에 start1, steam을 설치하고 steam을 통해 구매했던 게임들을 다시 설치해보았습니다...
3. 이메일
    1. 회사사람들 모두 아웃룩을 사용하고 있고, 저와 같이 이상한(?)사람은 썬더버드를 사용하고 있는데, 문서 규격와 글꼴이 맞지 않는 경우가 많아서 동료들이 좀 고생을 했었죠. 제를 거쳐서 전달되는 이메일은 하나같이 테이블이 이상한 위치에...ㅠㅠ
    그래서 저는 OS에 기본탑제된 에볼루션을 사용하고 있습니다.(에볼루션을 flatpak에서 설치한 경우 특정버전이 업그레이드 되면서 기존에 저장된 이메일을 찾지 못하는 경우가 아주 가끔, 정말 가끔 있었어요.)
4. 금융정보
    1. 리눅스에서는 뭐가 잘 안됩니다..ㅋㅋ 그래서 핸드폰으로 하고 있어요(!)

사이트신뢰성엔지니어링(SRE)의 요약

Jpub에서 발간한 사이트신뢰성 엔지니어링에 대해 읽고, 자체적으로 해석한 결론입니다. (정확하진 않을수 있습니다. 말 그대로 자체 해석...)

구매는 요기 - http://www.yes24.com/Product/Goods/57979286

SRE란? 구글에서 시스템의 안정성을 증가시키기 위해 활동하는 역활로, Devops보다 한단계 더 발전한 모델이라고 합니다.

SRE역활

1. Site Reliability Engineer 의 약자로, 소프트웨어 엔지니어링과 IT인프라 운영 그 중간쯔음에서 일하는 역활로, in-house tool이나 오픈소스를 활용해 시스템의 안정성과 확장성을 유지하고 개선하는 업무
2. Devops/SRE의 업무 목표는 신속한 서비스 제공을 위해 기업문화, 자동화툴을 이용한 플랫폼 설계/구축 하는 공통적인 업무영역이 있는데, 접근하는 방법이 살짝 다른듯. 어떠한 문제와 개선을 하기 위해서는 "무엇을 해야 하는지?-devops관점"과 "어떻게 할 수 있는지-SRE관점"의 차이점이 있는듯 하다.
3. 기본적으로 기존 운영팀이 처리하던 업무를 이어서 수행하게 되는데, 소프트웨어 엔지니어들이 모인팀이기 때문에 자동화된 소프트웨어를 설계 / 구현하는 팀이다. 다만 전화응대, 수작업, 티켓처리등의 기존 운영팀이 수행하던 업무는 최대 50%정도의 시간만 투입해야 자동화된 시스템 개발할 수 있는 여력을 확보할 수 있게 된다
    1. 운영성 업무를 수행할때는 업무시간에는 최대 2건의 업무만을 담당하게 하고, 업무를 배정받은 엔지니어는 정확 + 신속하게 업무를 수행하고 사후섬토=포스트모텀(Postmorterm)[1)] 을 작성하도록 한다. 2건이 넘어가는 경우 업무의 과중으로 문제점에 대한 관찰력이 저하되는것을 구글은 경험했다
4. 가용성, 응답시간, 성능, 효율성, 변화관리, 모니터링, 장애대응 그리고 수용량 계획에 대한 ownership을 가지게 되는데 이 부분은구체적인 사항은 조금 더 학습이 필요한 영역이다.

기업문화의 변화

1. 포스트모텀 작성시 특정절차에 대해 비난해서는 안되고 실수를 공유하여 스스로 고쳐나갈 수 있도록 유도해야 동일한 문제가 발생시 대응방안들의 절차들이 도출될 수 있다
2. 100% 신뢰성을 가질수 있는 시스템은 없다. 왜냐하면 사용자와 서비스간의 여러가지 요소들(사용자 단말, ISP, 가정의 인터넷, 전력 등)이 있기 때문에 이런 문제들을 해결하기 위해 에러 예산(Error budget)이라는 개념을 도입하였다. (앞으로 기술한 내용중의 에러예산에 대한 내용이 자주 나올예정이다.)
3. 시스템을 구성할때 목표 가용성을 설정해야 한다, 가용성 목표가 정의되면 에러 예산은 1에서 목표 가용성을 뺀 값으로 확보하면된다. 예를들어 목표 가용성이 99.99%라면, 에러예산은 0.01%인 셈이고, 이 에러예산을 초과하지 않는 범위내에서 엔지니어링 업무를 수행하면 된다.

에러예산의 활용방안

1. 구조화된 데이터를 위한 분산 저장소는 응답속도가 빨라야 하고, 높은 신뢰성을 제공해야 한다. 기 시스템을 어떤곳에서는 오프라인 분석을 정기적으로 수행하기 위한 저장소로 사용하기도 하고, 어떤팀은 신뢰성보다는 처리량을 중요하게 생각하는 곳도 있다.
   이런 활용방식을 모두 만족할 수 있는 방법중 하나는 모든 서비스에 엄청나게 높은 신뢰성을 제공할 수 있도록 개발하면 된다. 하지만, 실제로 구현하기에는 엄청난 비용이 발생하기 때문에 현실적으로 어려움이 있다.
2. 개발팀의 업무 목적은 새로운 기능을 출시하여 새로운 사용자를 확보하려고 한다. 때문에 SRE는 새로운 기능을 출시하기 위해 감수해야할 리스크를 활용하는데 사용하는것이 가장 이상적인 활용방안]
3. 목표 가용성 수준을 결정하기 위한 핵심 요소중의 하나가 비용.
    1. 서비스들을 구축하고 운영하는데 가용성 목표 상향시 수익에 긍정적으로 나타나는 효과는 무엇인지
    2. 목표 상향 후 예상되는 추가 수익이 투입된 비용을 상쇄시킬수 있는지. 예를들어 아래 수준으로 반영 할 수 있다.
        - 가용성 상향 목표치 : 99.9 % --> 99.99%, 향상되는 가용성은 0.09%
        - 해당 서비스에서 발생하는 수익 : 1,000,000,000원
        - 목표 상향을 위한 비용 : 1,000,000,000원 * (0.09 / 100) = 900,000원
        - 이 경우 가용성 수준을 0.09% 향상 시키는 비용이 90만원 이하라면 투자 가치가 있으나 90만원을 초과하는 경우라면 예상수익을 초과하기 때문에 목표 상향 가치를 재 검토해야 필요함

업무 내역

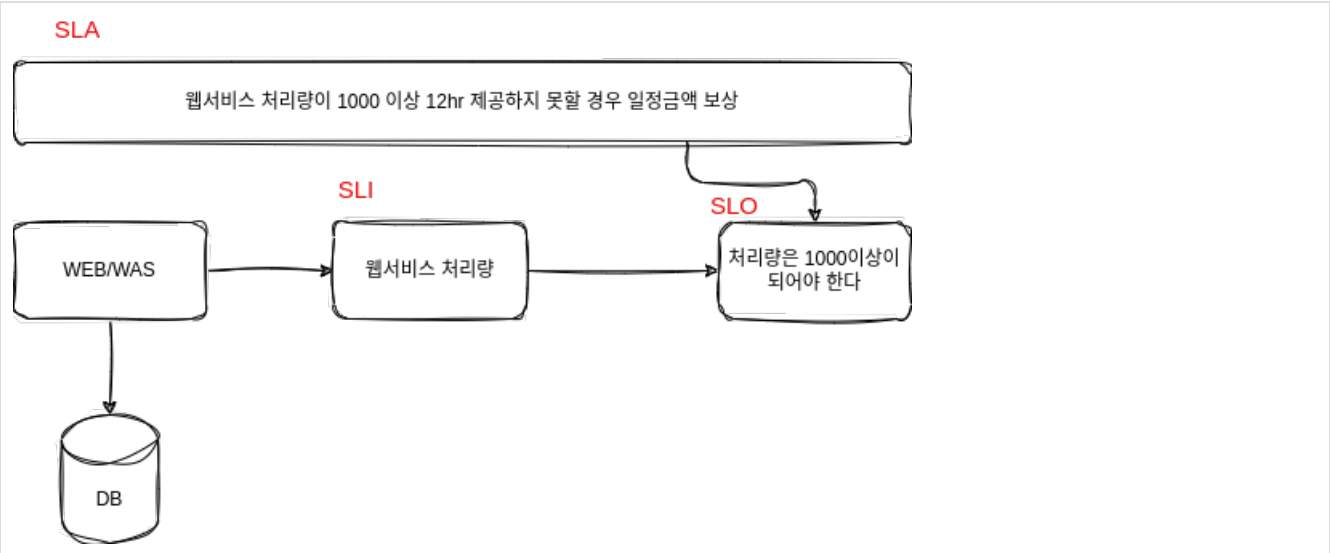
1. 모니터링
    1. 서비스의 소유자가 시스템의 상태와 가용성을 점검하고, 모니터링 전략이야기 말로 철저한 계획하에 수립되어야 하는 업무영역이다. 현재 대부분은 특정 임계치를 초과하거나 상태의 변화가 감지되면 noti하는 방식으로 처리하고 있는데, 이런 방법은 효과적인 해결방안이 아니다. 사람이 개입해서 판단하고 결정해야 하는 절차 자체가 문제가 있는 것이다.
    2. 신뢰성이란 문제가 발생하기전 동작했던 평균시간(MTTF)과 평균 수리시간(MTTR)을 의미한다. 이중에 긴급대응의 효율성을 나타내는 수치는 MTTR이 된다.
       사용자가 개입하여 장애를 처리하는것이 시스템이 직접 처리하는 경우보다 MTTR이 3배 이상 증가하는것으로 구글은 검증하였고 실력이

있는 엔지니어가 처리하는 절차도 뛰어나나 잘 만들어진 장애 대응 문서로 훈련된 엔지니어가 보다 더 수행절차가 더 뛰어났다. 때문에 엔지니어들의 장애상황에 대응할 수 있도록 훈련을 지속하고 있다.

2. 변화관리
    1. 구글 경험상 70%가량의 장애는 서비스 중인 시스템의 변화로 인한 문제로 제품의 단계적 출시, 문제발생을 빠르고 정확하게 도출하고 이전버전으로 롤백하는 절차들을 수행하면 장애상황의 최소화가 가능하다.
3. 수요예측과 계획수립
    1. 예측과 계획은 정확하지 않을수 있는 지표때문인지 상당수 수용력을 확보하기 위한 과정을 준비하지 않고 있다.
    2. 수용계획은 자연적 성장(사용자가 활용하면서 생기는 성장)과 인위적 성장(새로운 기능 추가) 모두 고려해야 한다,.

업무지표

1. 안정성과 확장성을 수행하기 위해서는 기준되는 지표가 필요한데, 여기서 나오는 용어가 SLA / SLI / SLO 3가지 용어
    1. SLA(Service Level Agreement)서비스 수준 협약 - 운영팀과 고객간의 서비스 수준에 대해 품질, 가용성 등 구체적인 기준을 설정하는 지표 (구글의 경우 법적인 효력이 있기 때문에 쉽게 변경해서는 안됨)
    2. SLI(Service Level Indicator), 서비스 수준 척도 - 서비스 수준을 측정하는 지표. 예를들어 대기시간, 가용성, 처리량 등의 자료가 포함
        1. SLI의 표준화 (일반적인 정의를 표준화 하기를 권장한다)
            1. 수집간격 : 평균 1분,
            2. 수집 범위 : 클러스터에서 수행되는 모든 tasks
            3. 측정빈도 : 매10초
            4. 집계에 포함할 요청들 : 블랙박스 모니터링으로 수집한 HTTP(GET요청들)
            5. 데이터 수집방식 : 모니터링시스템에 의해 서버에서 수집
            6. 데이터 액세스 응답시간 : 마지막 바이트가 전송된 시간
    3. SLO(Service Level Objective), 서비스 수준 목표 - SLI에서 도출된 지표를 어느정도의 수준으로 품질을 정할것인지 정하는 기준, 발생 가능한 위험과 실행가능성에 대해 조언할 수 있어야 하며, 구글에서는 몇가지 도출안을 제시하였다.
        1. 목표치 설정 가이드
            1. **현재 성능을 기준으로 목표치를 설정하지 말것** : 시스템의 장점과 한계를 이해하고 있어야 하며, 높은 시스템의 이해도를 바탕으로 목표치를 설정해야 하는데 이를 고려하지 않을 경우 시스템에 엄청난 노력을 투입하고, 재설계 없이는 시스템 향상이 불가능하게 될 수도 있게 된다
            2. **최대한 단순하게 생각할것** : SLI를 복잡하게 집계하면 시스템 성능 변화를 명확하게 반영하지 못하고 그 원인 파악이 어렵게 된다.
            3. **자기 만족에 얽매이지 말것** : 응답시간의 저하없이 시스템 부하를 확장하는 것은 매력적인데 사실상 이런 요구는 현실성이 없다.그런 시스템을 설계하게 되면 매우 긴 기간을 추가로 투입되어야 하며 운영비용도 많이 발생하게 된다.
            4. **가능한 적은 수의 SLO를 설정할것** : 시스템의 특성을 확인할 수 있는 최소한의 SLO를 선택해야 한다.
            5. **처음부터 완벽하게 설정은 불가능하다** : SLO 정의와 목표는 시간이 지남에 따라 시스템의 동작을 살피면서 언제든지 다시 정의할 수 있어야 한다. 처음부터 지나치게 높은 목표를 설정하고 이 목표치를 완화하는것보다 처음부터 조금은 느슨한 목표를 설정한 이후에 조금씩 강화하는것이 낫다.
        2. 기대치 설정 가이드
            1. 안전 제한선을 지킬것 : 대고객 SLO보다는 좀 더 보수적인 값으로 설정된 SLO를 지키면 만성적인 문제들이 외부로 노출되기 전 적절하게 대응할 수 있는 여력이 필요하다
            2. 지나친 목표 설정하지 말것 : 서비스층 확보는 실제 사용자들이 경험한 것에 의해 이루어진다.
2. 대충이렇게 정리할 수 있을듯 하다.



위험 요소 관리

1. 기회비용 : 위험에 대비하기 위한 시스템이나 기능을 구현하기 위해 투입할때 발생하는 비용을 의미하며 엔지니어들은 사용자를 위한 새로운 기능과 제품 개발업무 수행이 불가능하다.
2. 서비스중 발생 가능한 위험 요소를 찾아내는 활동을 수행하는데 필요 이상의 신뢰성을 확보하는 활동을 수행하지 않는다. 즉. 목표치가 99.99% 인 시스템을 그 이상으로 초과달성하려고 노력을 하나 넘치게 초과하는 활동은 하지 않는다. 넘치게 초과하기 위한 활동이 이루어질 경우 시스템에 새로운 기능을 추가하거나 운영 비용을 줄기이 위한 기회를 잃어버리기 때문.
3. 서비스 가용성을 판단하는 가장 직관적인 방안은 의도치 않은 장애로 인한 다운타임을 측정하는데, 시간을 기준으로 한 가용성 공식은 다음과 같다. (계획된 서비스 다운타임은 장애라고 판단하지 않는다)

   $$\text{가용성} = \frac{\text{업타임}}{(\text{업타입}+\text{다운타임})}$$

   이 공식을 계산하면 99.99%인 경우 연간 가능한 다운타임은 52.56분 이다. 다만, 멀티AZ와 같이 전세계로 분산된 서비스를 운영하는 시스템의 경우 요청성공율에 기초한 가용성을 정의하는것이 효과적이다.

   $$\text{가용성} = \frac{\text{성공한 요청수}}{\text{전체요청수}}$$

   이 공식을 대입하면 하루에 250만개 요청을 청리하는 시스템은 하루에 250개 이내인 경우 99.99%를 달성할 수 있다.
4. 서비스 위험 수용도를 결정하기 위한 요소
   1. 어느정도 수준의 위험 수용도가 요구되는지
   2. 장애 종유에 따라 서비스에 미치는 영향이 달라지는지
   3. 지속적으로 발생하는 위험 중 어느지점에 서비스 비용을 투입할것인지
   4. 중요하게 고려해야할 서비스 지표는 어떤것들이 있는지
5. 구글에서 정의한 목표 가용성 수준 정의
   1. 사용자는 어느정도 수준의 서비스를 기대하는지
   2. 수익과 직적접인 연관이 있는지
   3. 유료서비스인지, 무료서비스인지
   4. 시장에 경쟁자가 있다면 어느정도 수준의 가용성을 제공하는지
   5. 대상이 개인 고객인지, 기업 고객인지 (기업고고객의 가용성 목표가 조금 더 높다)

Reference

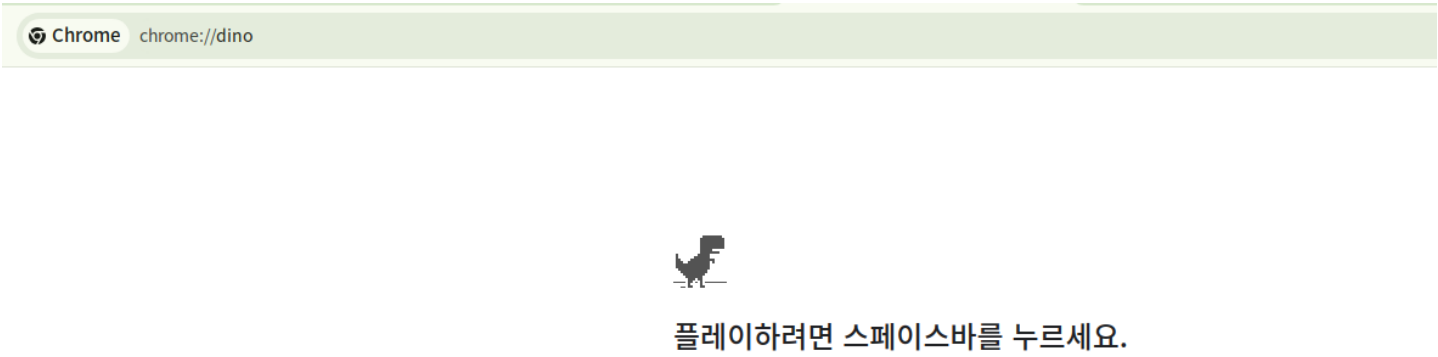
- 사후검토(Postmoterm) - https://scienceon.kisti.re.kr/srch/selectPORSrchReport.do?cn=KAR2005016666

크롬브라우져에서 공룡게임하기

크롬브라우져에 공룡게임 있다는 사실을 알게되었는데.....

https://www.youtube.com/embed/J1uxbybhv-M

크롬 브라우져에서 chrome://dino를 입력하면 인터넷 연결을 끊지 않아도 할 수 있어요.ㅋㅋ



그리고 이건..ㅋㅋ 내 점수..ㅋㅋ

관측 가능성 (Obserability) 요약

⚠ 개인적으로 공부했던 부분이라 사실과 다르게 자의적인 해석이 있을수 있음

왜 옵저빌리티(observability)를 도입해야 하는가?

1. 서비스 장애시 기존에는 담당자의 경험으로 처리했으나, 컨테이너화 되면서 내부시스템은 복잡해졌고, 단순 모니터링으로는 이슈대응이 매우 어렵게 되었음
2. 목표는 내부시스템에 대한 자세한 정보를 기반으로 미래에 발생할 이벤트를 예측하고, 이런한 예측을 바탕으로 IT운영을 자동화 하는 것.
   (ef. 장애가 발생할 위험이 있으면 미리 예측하고 운영자에게 통보, 서비스에 필요한 리소스 증감을 미리 예측하는 것)
3. 단순 모니터링이 아닌 서비스간의 상관관계를 파악하고 상태 데이터를 기사화 함으로써 개발자, 운영자에게 빠른 의사결정을 할 수 있도록 도와주기 때문에 필요하다.
4. 옵저빌리티와 AI를 결합한 AIOps로 나아가기 위한 발판마련

옵저빌리티 원론

1. 옵저빌리티의 한줄요약

   > “ 오직 시스템의 외부 출력만을 이요해 시스템의 형재 상태를 이해할 수 있는 능력

2. 옵저빌리티를 구성하기 위한 조건
   - 어플리케이션의 내부 동작 이해
   - 모든 시스템 상태를 이해함
   - 외부 툴을 이요한 관측데이터로 내부 동작과 시스템 상태 이해
   - 시스템 상태의 상태와 상관없이 이해할수 있어야 함
3. MELT(Metric / Event / Log / Trace)기능을 포함하여 흩어진 데이터를 한곳에 모아 지속적인 관측을 통해 서비스 이슈 발생시 MELT를 이용해 분석, 해결할 수 있는 방법을 제공
4. 옵저빌리티는 특정 Tool의 집합체가 아닌 개념원리로 접근이 필요할듯

옵저빌리티 사용 용어

1. 카디널리티(cardinality) - 유니크한 값의 수 (수치가 낮다 - 데이터의 중복도가 높다, 수치가 높다 - 데이터의 중복도가 낮다.) 주민등록번호 (유니크한값이 있고 중복데이터가 없기 때문에 카디널리티가 높다라고 표현
2. 화이트박스 모니터링
   - 로그나 프로파링일 인터페이스를 제공하는 핸들러를 이용해 내부 측정 기준에 따라 모니터링 의미. http 응답코드 500 에러 발생횟수, 레이턴시 등. 현상이 발생한 근거를 확인하기 위한 방식
3. 블랙박스 모니터링
   - 사용자가 보게되는 기능을 외부에서 테스트 하는 방법 인프라 수준의 모니터링에 유용, 클러스터 정상작동 여부같은 것들 모니터링하기 때문에 디버깅 정보로 활용은 어려움.
   - 서버상에 나타나는 증상을 기본으로 하며 발생하는 현재 상황에 대해 모니터링
4. 옵저빌리티의 영역에서는 화이트 박스를 기반으로 예측하여, 모니터링 영역에서는 블랙박스 기반으로 진행한다.
5. 모니터링의 기본개념은 모든것을 수집하고 모니터링하기 보다는 적은 데이터라도 의미있고 예측 가능한 결과를 도출해 내는것이 좋은 방법
6. 전통적인 모니터링은 블랙박스를 기본으로 핵심적인 App과 시스템 메트릭을 중점으로 보는 반면, 옵저빌리티는 이벤트, 대그, 로그등을 결합해서 서비스에 대한 정보를 제공하는 것이 목표. 때문에 세부적인 시스템 내용을 포함하고 있으므로, 디버깅에 적합하다.

구성요소

1. 메트릭(Metric)
   - 일정시간동안 수집된 데이터를 집계하고 수치화하는것.
   - 전체적인 시스템의 상태를 확인할때 유용
2. 로그(Log)
   - 로그관리의 목적은 시스템에 분산된 로그를 한곳에 저장해서 다루기 쉽게 하는것.
   - 로그가 표준화되어 있지 않은 경우 쉽게 검색이 불가능하고 후속처리를 위한 데이터 가공을 어렵게 함. 때문에 로그 표준화가 미리 되어 있어야 하고, 언제 발생하게 될지 모르는 장애를 대비에 지속적으로 로그 수집 / 관리가 되어야 한다
   - 로그관리 시스템은 동적인 시스템으로 구성되어야 급격이 증가하는 로그를 받을수 있다
   - 서로 다른 데이터가 정확하게 결합이 될때 데이터의 활용가치를 높일수 있다. (로그/메트리/추적/트래픽 데이터는 이기종 데이터가 아니며 결합되지 않은 데이터일뿐이다)
3. 이벤트(Event), Newrelic / Redhat에서 제안 ** - 이벤트는 로그와 다르다 **
   - 중요한 분석지점의 세부적인 기록이며, 정확한 문맥 제공을 위해 첨부되는 메타데이터에서 발생한 중요사건을 기록한것
4. 추적(Trace)
   - 서비스가 시스템을 경유하며 트랜잭션을 처리하는 과정에서 발생하는 세부적인 정보
   - 대기시간, 지연, 병목이나 에러를 일으키는 원인을 문맥과 로그, 태그등의 메타에 출력한다.

기존 모니터링 솔루션과의 비교

역활의 차이점

1. 모니터링은 정해진 템플릿에 대해서 보여주는거 것. 무엇이 언제 일어났는지에 대해 초점이 맞춰져 있다면, 옵저빌리티는 왜 어떻게 일어났는가까지 파악할 수 있도록 도와준다.
2. 옵버저빌리티를 통해 평균해결시간(MTTR)을 가속화 한다
3. 옵버저빌리티는 모니터링을 대신하는것이 아님. (옵저빌리티가 모니터링보다 더 큰 영역의 업무범위)
4. 시스템에서 외부로 출력되는 값을 사용하여 시스템 내부 상태를 이해하고 예측하는 행위
5. 레거시 모니터링 솔루션의 정적 데이터를 대체하여 인프라의 전체 스펙트럼을 분석

기능의 비교

1. 모니터링
   - 이슈 발생시 수집된 데이터를 가지고 분석
   - 대시보드에 표현된 항목을 기반으로 한 해결방법 접근
   - 큰 변화가 없는 환경에서 시스템의 이해도가 높은 시스템 개발자가 주로 사용
2. 옵저빌리티
   - 이벤트 발생시 단순 메트릭외에 여러가지 데이터를 제공하여 원인 분석
   - 가설을 기반으로 한 해결방법 접근
   - 가변데이터가 많은 복잡한 인프라 환경에서 에서 시스템 이해도가 낮은 사람도 분석 가능

APM/DPM과 차이점

1. 옵저빌리티는 APM/DPM지표가 포함되지만, 로그, 이벤트, 추적과 같은 유형으로 확대하며 시스템 동작에 대해 전반적인 이해를 높이는데 목적이 있다
2. APM/DPM은 어플리케이션 성능 모니터링 및 관리 (응답시간, 오류율, 트랜잭션 추적 등)하며, UX과정에서 발생할수 있는 문제를 찾아내어 해결하도록 설계
3. APM/DPM은 특정 세션(어플리케이션 성능)에 초첨을 맞춘 옵저빌리티의 하위집합 개념이며, 옵저빌리티는 전체시스템 동작에 중점을 맞추었

음.

Tool

구축업체

옵저빌리티를 활용한 솔루션 구축업체는 생각보다 굉장히 많음, Whatap, New Relic, Elastic, IBM(Redhat), ServiceNow, 등등등

Opentelemtry

- ovservability의 통합 표준 OSS, 보편화된 가능성이 높음
- 리시버 : 데이터 수집(prometheus)
- 프로세서 : 데이터 정제(배치/필터 실행)
- 익스포터 : 백엔드로 데이터 전달(데이터독, E/S)
- 2022년 GA발표이후 벤더(데이터독, 뉴렐릭 등)에서도 자사 API보다 오픈텔레메트리API를 더 권장하고 있음

Appendix#1. 장애조치 관련 용어

1. MTBF(Mean Time Between Failures) : 평균 장애 간격, 예정된 중지시간은 계산하지 않음. 예상치 못한 서비스 중단이슈만 계산
   계산기간의 총 가동시간을 장애 발생 횟수로 나눔. ef. 24시간 동안 2시간의 가동중지의 경우 MTBF는 11시간 (24-2)/2(횟수)
2. MTTR(Mean Time To Repair) : 해결에 걸리는 평균시간, 수리하는데 걸리는 평균시간, 수리및 테스트 시간모두 포함하며, 완전시 작동할때까지의 시간을 측정
   수리에 소요된 총 시간을 합산 / 수리횟수 10회의 중단, 4시간 동안 수리했다면 240(분)/10(중단횟수) = 24분 , MTTR는 24분
3. MTTA(Mean Time To Acknowledge) : 평균확인시간, 알람이 트리거 된 시점부터 이슈작업이 시작되는 시점까지 평균시간
   알람과 확인사이의 시가늘 합산 인시던트 수로 나눔. 10개의 인시던트 발생, 10개의 알람과 대응시작 사이의 시간이 총 40분인 경우 40/10=4, 평균 4분 소요
4. MTTD(Mean Time to Detect) : 평균 탐지 시간, 문제 발생후 식별까지 소요되는 시간
5. MTTF(Mean Time To Failure) : 평균 장애시간, 수리 불가능한 장애사이의 평균시간을 의미하며 총 작동시간 합산 후 장치수로 나누기.

- MTTA와 MTTD의 차이는 MTTD는 이슈 발생 후 알람 수신까지 소요되는 시간이고, MTTA는 이슈 발생시 인지 후 작업 시작까지 소요되는 시간을 측정하는 차이 존재

Appendix #2. 프로메테우스 데이터

1. 데이터 관리방법
   - 프로메테우스는 TSDB(Time Series DataBase)유형의 데이터베이스이며, LRU알고리즘을 사용하는데, 가장 오랫동안 참조하지 않은 페이지를 교체
   - 메모리 페이징을 사용하는데, 일정크기의 페이지로 분한해서 메모리에 적재, 데이터를 수집하고 블록형태로 만들어 디스크에 저장함
   - 데이터대신 청크(Chunk)라는 용어를 사용, 다수의 청크를 포함해서 인덱스와 기타 데이터로 수정된것이 블록
   - 데이터셋은 다수의 데이터 그룹을 의미
   - 데이터 포인트는 대시보드에서 시계열로 출력되는 개별데이터
2. 데이터 경로 hierarchical

```
./data
 -abc
 -def                # 블록정보
  - chunks          # 청크경로
    - 1
  - tombstones  # 삭제 여부 표시파일
  - index           # 색인을 위한 레이블과 시간정보 인덱스 파일
```

```
  - meta.json     # 블록의 메타데이터
- checks_head   # 청크 헤드
  - 1
- wal
  - 001
    - checkpoint.001   # wal 파일
      - 000                 # 복구를 위한 체크포인트 wal파일
```

- WAL(로그 선행기록, Write Ahead Logging)은 장애 복구를 위해 사용, 프로메테우스는 메모리에 우선 저장하고 주기적으로 flush하면서 디스크에 청크를 생성하는데, 청크를 생성하기전 메모리에 있는 데이터를 백업하는 용도로 WAL생성, 프로메테우스 장애시 메모리에 있는 데이터에 문제가 생기며 WAL을 사용헤 메모리에서 관리하는 데이터를 복구 수행

1. 관련옵션
   - storage.tsdb.min-block-duration : 하나의 블록에 저장된 데이터의 시간, 2h로 선언하는 경우 하나의 블록디렉토리에는 두시간의 데이터 저장
   - storage.tsdb.max-block-duration : 하나의 블록에 최대로 저장할 수 있는 시간, 기본값은 유지율을 설정하는 옵션인 storage.tsdb.retention.time의 10%값이다
2. 데이터 저장단계
   - 메모리에는 최대 두시간 데이터가 보관

Appendix#3. 기타 정보

1. 가트너에서 정의한 3가지
   - 모니터링 : 임계값 감시, 알람 발송
   - 옵저빌리티 : 어플리케이션의 혁신을 가속화 하는 프로세스로 모니터링이 발전한것
2. 옵버질리티 툴 : 개발자가 엔지니어가 예사치 못한 시스템 동작을 보다 효과적으로 설명이 가능. 이중 opentelemetry가 데이터를 수집하는 역활을 수행하는것으로 보임. 다만 로그 수집을 하는게 아니라서 로그수집은 ekf stack으로 분할하거나 이벤트 데이터를 efk stack으로 모으는 방법도 가능할듯

옵저빌리티를 완성하기 위한 여정

1. 관련된 Tool 도입전 협업
2. 로그의 표준화 설정 필요

Reference

- 에쓰씨케이
- Elastic
- New Relic
- 이글루코퍼레이션
- IBM
- ServiceNow
- Redhat
- 와탭블로그