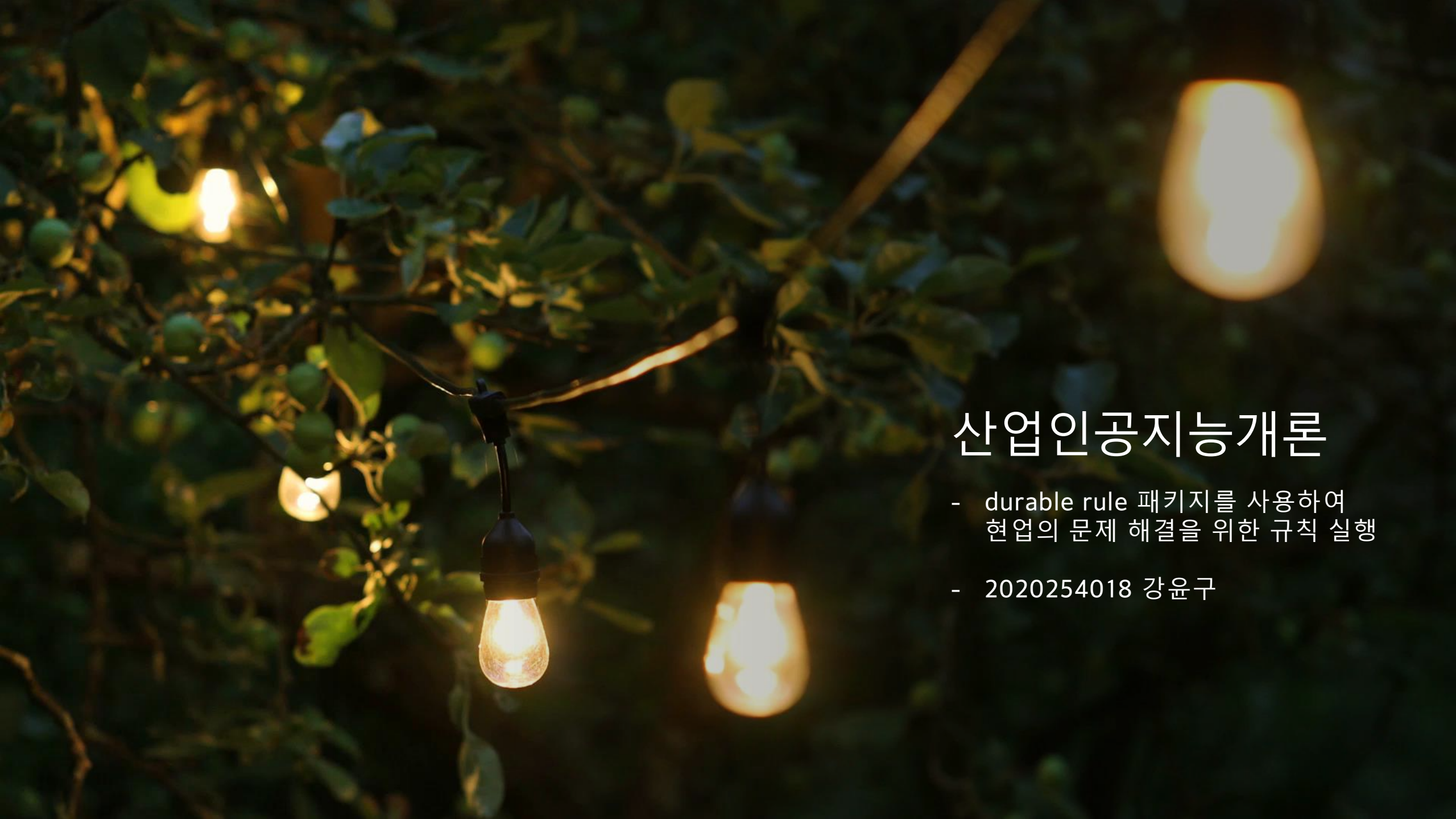


산업인공지능개론

- durable rule 패키지를 사용하여
  현업의 문제 해결을 위한 규칙 실행

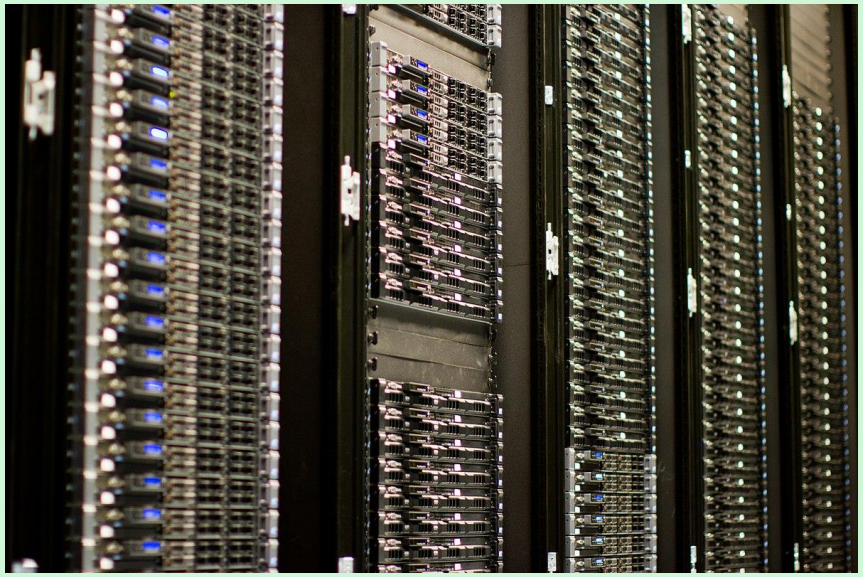
- 2020254018 강윤구

1. 현업의 문제

서버 통합 유지보수 업무를 처리하다 보면 다음과 같은 유형의 장애가 발행하여 고객과 상담을 하는 경우가 발생

1) 전원 장애
- 상황 : 서버의 전원이 안 켜진다.
- 확인 절차
  (1) 전원코드 연결이 잘 되어 있는지 확인
  (2) Power Supply가 불량인지 확인

2) 통신 장애
- 상황 : 서버와 통신이 되지 않는다.
- 확인 절차
  (1) Cable 연결이 잘 되어 있는지 확인
  (2) Port LED 상태를 확인
  (3) IP 주소가 잘 설정 되어 있는지 확인
  (4) Ping Test 를 통한 통신 상태 확인



1. 현업의 문제(계속)

3) 입력 장애
- 상황 : 키보드, 마우스 등이 안된다.
- 확인 절차
  (1) 입력 장치를 재연결 하고 확인
  (2) 원격접속 여부 확인
  (3) 서버 Hang 상태이면 강제로 재기동

4) 처리 속도 지연
- 상황 : 서버가 느리다.
- 확인 절차
  (1) Process중 자원 점유율이 높은 서비스 확인
  (2) 자원 점유율이 높은 서비스 Kill
  (3) 서비스 데몬을 재시작

5) Booting 장애
- 상황 : OS 부팅이 안된다.
- 확인 절차
  (1) OS Disk 상태 확인(황색 LED가 점등된 Disk 가 있는지 확인)
  (2) Single 부팅을 시도해 본다.
  (3) Backup 본으로 복구를 시도한다.

2. durable rule 패키지를 사용하여 규칙 실행

```
from durable.lang import *

with ruleset('Machine'):
    @when_all((m.predicate == '안켜진다') & (m.object == '전원이'))
    def check1(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '확인한다', 'object': '전원코드를' })

    @when_all((m.predicate == '안켜진다') & (m.object == '전원이'))
    def check2(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '확인한다', 'object': '파워서플라이를' })

    @when_all((m.predicate == '안된다') & (m.object == '통신이'))
    def check3(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '확인한다', 'object': 'Cable을' })

    @when_all((m.predicate == '안된다') & (m.object == '통신이'))
    def check4(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '확인한다', 'object': 'Port의 LED를' })

    @when_all((m.predicate == '안된다') & (m.object == '통신이'))
    def check5(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '확인한다', 'object': 'IP를' })
```

```
    @when_all((m.predicate == '안된다') & (m.object == '통신이'))
    def check6(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '해본다', 'object': 'Ping테스트를' })

    @when_all((m.predicate == '안된다') & (m.object == '입력이'))
    def check7(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '재연결해본다', 'object': '입력장치를' })

    @when_all((m.predicate == '안된다') & (m.object == '입력이'))
    def check8(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '해본다', 'object': '원격접속을' })

    @when_all((m.predicate == '안된다') & (m.object == '입력이'))
    def check9(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '재기동한다', 'object': '강제로' })

    @when_all((m.predicate == '느리다') & (m.object == '속도가'))
    def check10(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': '체크한다', 'object': 'Process를' })

    @when_all((m.predicate == '느리다') & (m.object == '속도가'))
    def check11(c):
        c.assert_fact({ 'subject': c.m.subject, 'predicate': 'Kill한다', 'object': '사용량이 많은 Process를' })
```

```
@when_all((m.predicate == '느리다') & (m.object == '속도가'))
  def check12(c):
    c.assert_fact({ 'subject': c.m.subject, 'predicate': '재기동한다', 'object': '데몬을' })

  @when_all((m.predicate == '안된다') & (m.object == '부팅이'))
  def check13(c):
    c.assert_fact({ 'subject': c.m.subject, 'predicate': '확인한다', 'object': 'DISK의 LED를' })

  @when_all((m.predicate == '안된다') & (m.object == '부팅이'))
  def check14(c):
    c.assert_fact({ 'subject': c.m.subject, 'predicate': '해본다', 'object': 'Single 부팅을' })

  @when_all((m.predicate == '안된다') & (m.object == '부팅이'))
  def check15(c):
    c.assert_fact({ 'subject': c.m.subject, 'predicate': '복구한다', 'object': 'Backup본으로' })

  @when_all(+m.subject)
  def output(c):
    print('Fact: {0} {1} {2}'.format(c.m.subject, c.m.object, c.m.predicate, ))


assert_fact('Machine', { 'subject': 'Server', 'object': '입력이', 'predicate': '안된다' })
assert_fact('Machine', { 'subject': 'Service', 'object': '속도가', 'predicate': '느리다' })
assert_fact('Machine', { 'subject': 'Server', 'object': '전원이', 'predicate': '안켜진다' })
assert_fact('Machine', { 'subject': 'OS', 'object': '부팅이', 'predicate': '안된다' })
assert_fact('Machine', { 'subject': 'Server', 'object': '통신이', 'predicate': '안된다' })
```

3. 결과

```
Fact: Server 강제로 재기동한다
Fact: Server 원격접속을 해본다
Fact: Server 입력장치를 재연결해본다
Fact: Server 입력이 안된다
Fact: Service 데몬을 재기동한다
Fact: Service 사용량이 많은 Process를 Kill한다
Fact: Service Process를 체크한다
Fact: Service 속도가 느리다
Fact: Server 파워서플라이를 확인한다
Fact: Server 전원코드를 확인한다
Fact: Server 전원이 안켜진다
Fact: OS Backup본으로 복구한다
Fact: OS Single 부팅을 해본다
Fact: OS DISK의 LED를 확인한다
Fact: OS 부팅이 안된다
Fact: Server Ping테스트를 해본다
Fact: Server IP를 확인한다
Fact: Server Port의 LED를 확인한다
Fact: Server Cable을 확인한다
Fact: Server 통신이 안된다
Out[1]:
{'$s': 1, 'id': 'sid-0', 'sid': '0'}
```