Daily Continuous Deployment를 위한 Custom CLI 개발 및 AWS Elastic Beanstalk에 적용하기

2016. Aug. 13.

한종원 (addnull@gmail.com)

PYCON APAC 2016

Kanizsa Lab

발표자 소개

PYCON APAC 2016

Kanizsa Lab

- 이름: 한종원 (경기도 안양 거주)
- '의미가 있는 일을, 올바르게 하고 싶다.'라는 생각으로 스타트업 시작
- 'Elastic DevOps Engineer'
  Python, AWS 환경에서 유연하고(scalability), 장애 복구가 쉬우며 (fail-over), 지속적인 개발 통합(CI)되는 backend infra 개발, 운영
- 현재 택시 빈자리 공유 서비스 '캐빗(Cabbit)'을 만드는 '카니자랩(Kanizsa Lab)'에 backend infra 전반을 담당

발표 내용

PYCON APAC 2016

Kanizsa Lab

- 'Daily Continuous Deployment' (이하 'daily CD') 소개 및 장단점
- 요구 사항들
- GitHub repo 'Nova', 'Johanna' 소개 및 Live Demo
- Custom CLI 개발
- 지난 6달간 실제 서비스로 운영 경험
- Wrap Up & QnA

발표 내용

PYCON APAC 2016

Kanizsa Lab

- 40장이 넘는 발표 자료 ...

'Daily CD' 소개 및 장단점

PYCON APAC 2016

Kanizsa Lab

Daily CD란?

- from Wikipedia
  "**Continuous delivery** is sometimes confused with **continuous deployment**. Continuous deployment means that every change is automatically deployed to production."

- 정의: 매일 자동으로 최신 버전의 소스 코드를 테스트, 패키징, 배포

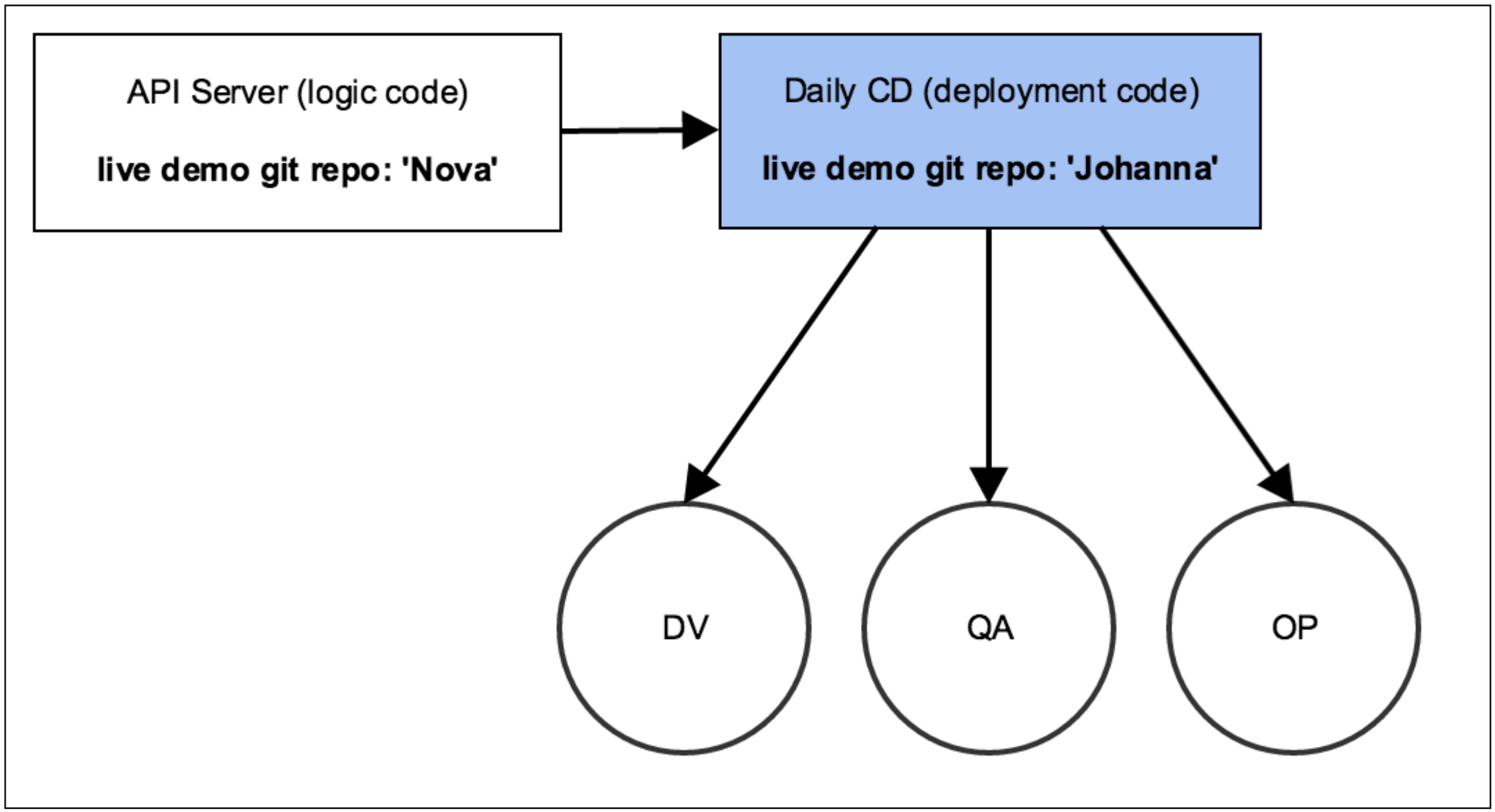
- 현재 구축 운영하는 시스템 현황
  - Server & web client # 실제 배포(deployment)
  - Android app # continuous delivery까지만 진행
  - iOS app # 아직 미적용 (즉, 사람 손으로...)
- 오늘 발표 내용 범위는 'server'에 한정

Daily CD란? (cont.)

PYCON APAC 2016

Kanizsa Lab

- 실제 server/client의 logic code와 deployment code를 분리



Daily CD 장단점

PYCON APAC 2016

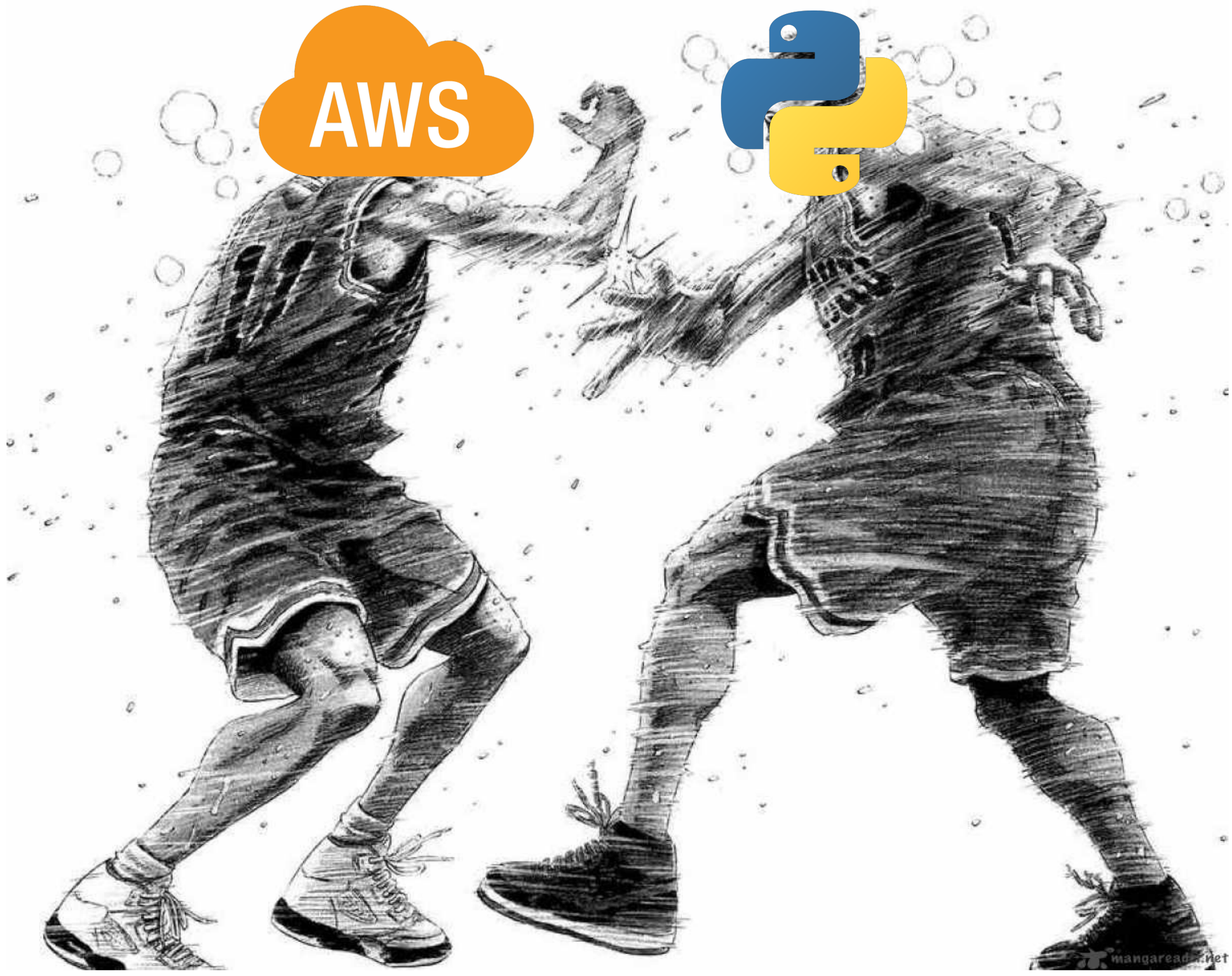
Kanizsa Lab

- 단점
  - 'Daily CD'는 "독립된 하나의 시스템"
  - **추가 개발**이 필요하며, **추가 유지 보수** 대상이 생김
  - 기존에 legacy system(daily CD가 없는)에 적용하기 거의 불가능하거나 큰 비용이 요구될 수 있음.
    -> 즉, 첫번째 단추를 잘...
- 결국 모두 다 추가 **비용** -> 경영진의 설득이 필요

Daily CD 장단점 (cont.)

- 장점 (with AWS cloud infra)
  - 'human error'를 줄일 수 있다. (<- 자동화의 장점)
  - 한번 잘 만들어 두면, 재사용하기 쉽다.
    즉, backend infra의 양산화(DV, QA, OP) 가능
  - Service/Software의 lifetime이 길어져도 개발, 테스트, 배포 시간을 거의 동일하게 유지
  - 새로 입사한 개발자에게 바로 개발 환경을 제공 가능
  - 별도의 QA 환경을 원하는 때에만 운영이 가능
    -> 즉, 비용 절감

Daily CD 장단점 (cont.)



요구 사항들

PYCON APAC 2016

Kanizsa Lab

요구 사항들

- 'no service downtime'
- 추가 비용의 최소화
- 문제 발생 시 rollback 또는 hot fix 전략
- log, data 유실 방지
- 사용할 software stack (OS, web server, DB engine, PL 등)
- 'notification' (즉, 'monitoring/alarm')

요구 사항들에 대한 답변

PYCON APAC 2016

Kanizsa Lab

- 'no service downtime'
  -> AWS Elastic Beanstalk(이하 'EB')의 CNAME swapping

- 추가 비용의 최소화
  -> AWS on-demand resource 사용

- 문제 발생 시 rollback 또는 hot fix 전략
  -> rollback은 없다. 최소 시간(10분 이내)으로 hot fix를 반영한다.

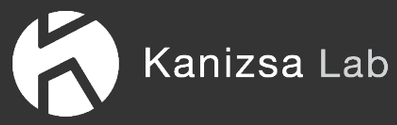
- log, data 유실 방지
  -> AWS EB의 15분 주기의 log rotation 및 S3 upload

요구 사항들에 대한 답변 (cont.)

- 사용할 software stack (OS, web server, DB engine, PL 등)
  -> Amazon Linux 및 AWS EB 기본 설정
  -> RDS 'mysql', python

- 'notification' (즉, 'monitoring/alarm')
  -> AWS CloudWatch + Zabbix
  -> Atlassian Bamboo (continuous integration server)
  -> HipChat / Slack (REST API call)

GitHub repo 'Nova', 'Johanna' 소개 및 Live Demo



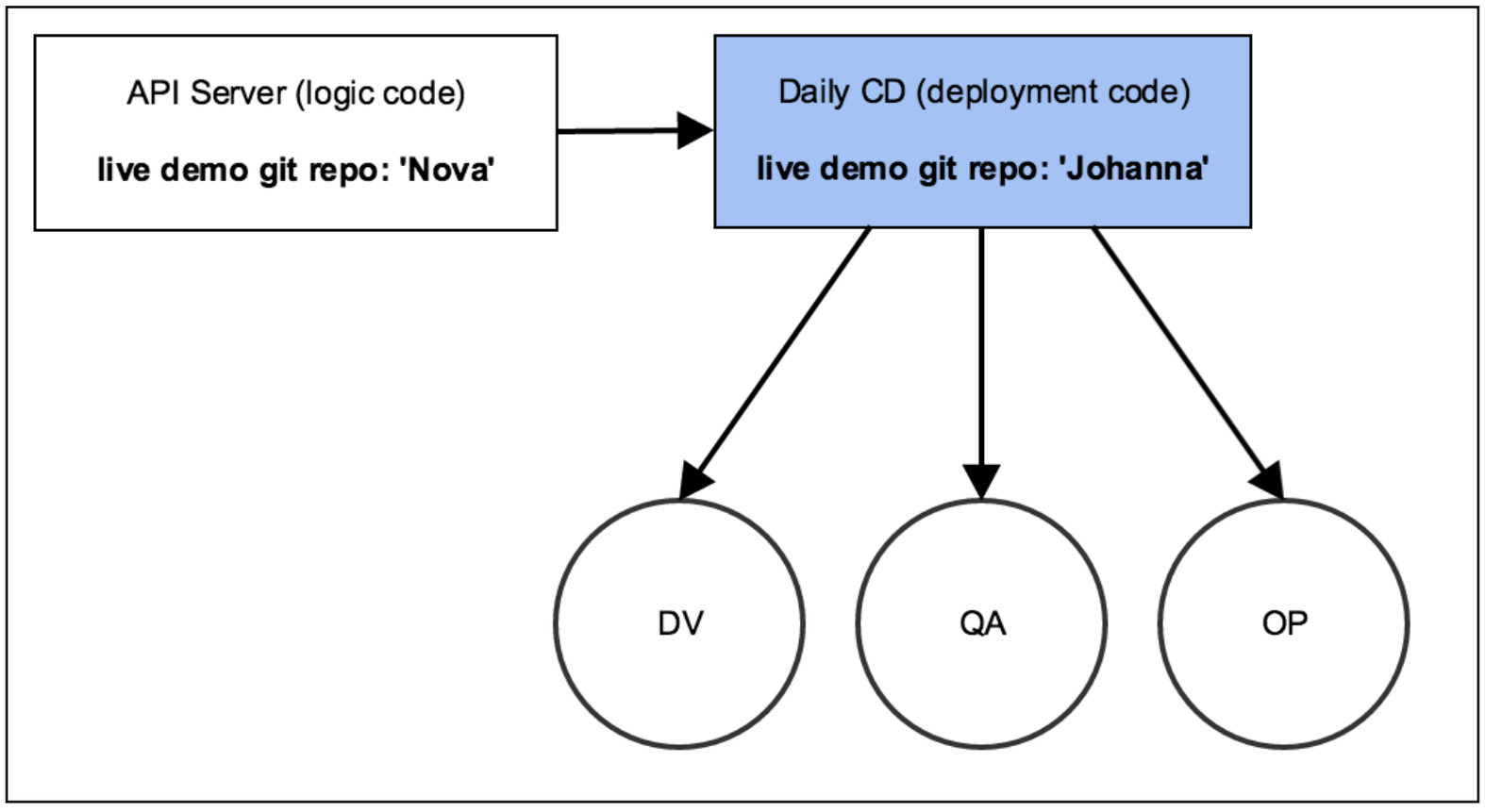
Live Demo 내용 소개

PYCON APAC 2016

Kanizsa Lab

- Demo의 범위는 'backend infra'로 한정
  (Django 기반 API server 'Nova' 1 set 포함)

- 발표에 사용된 Custom CLI의 source code의 GitHub repo
  - https://github.com/addnull/nova
  - https://github.com/addnull/johanna/tree/0.0.1



Live Demo 내용 소개 (cont.)

PYCON APAC 2016

Kanizsa Lab

- https://github.com/addnull/nova
  - REST API server with Django. (OpenStack의 'Nova'가 아님)
  - Daily CD demo를 위한 sample -> API는 '(GET) /echo/'만 있음
- https://github.com/addnull/johanna/tree/0.0.1
  - 실제 **(re)provisioning**과 **daily CD**를 위한 python script
  - 'provisioning'이란?
    아무것도 없는 맨바닥에서 시작해서 실제 서비스 운영 가능한 세트를 구축하는 일.

Live Demo 내용 소개 (cont.)

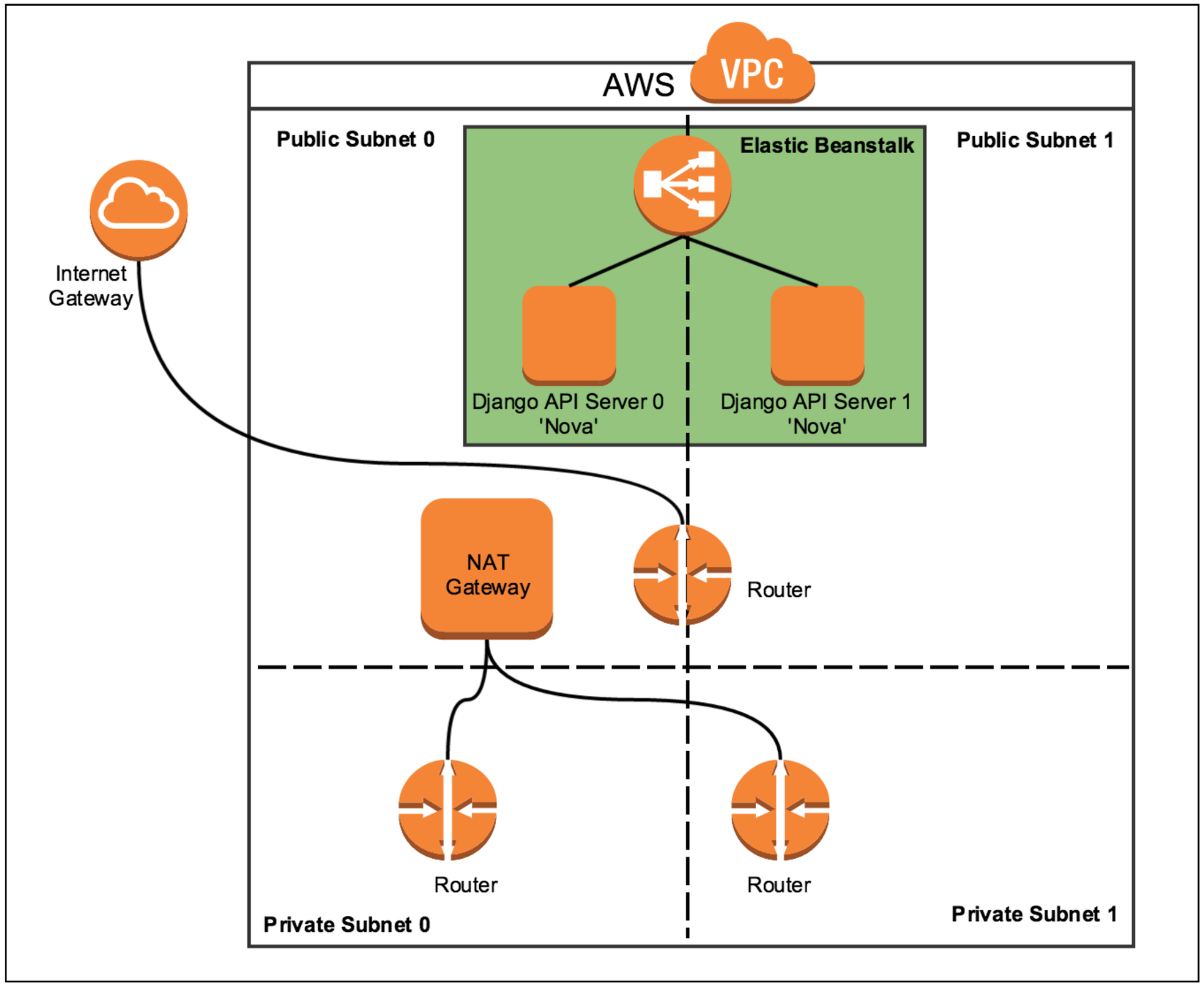
- Live demo 용 backend infra 전체 구조
  - Network
    - VPC
    - public/private subnet
    - public/private route table
    - internet gateway (for public subnet)
    - nat gateway (for private subnet)
- VM (Elastic Beanstalk Environment)
  - ELB, EC2 (<- 실제 daily CD 대상)

Live Demo 내용 소개 (cont.)

PYCON APAC 2016

Kanizsa Lab

- Live demo 용 backend infra 전체 구조

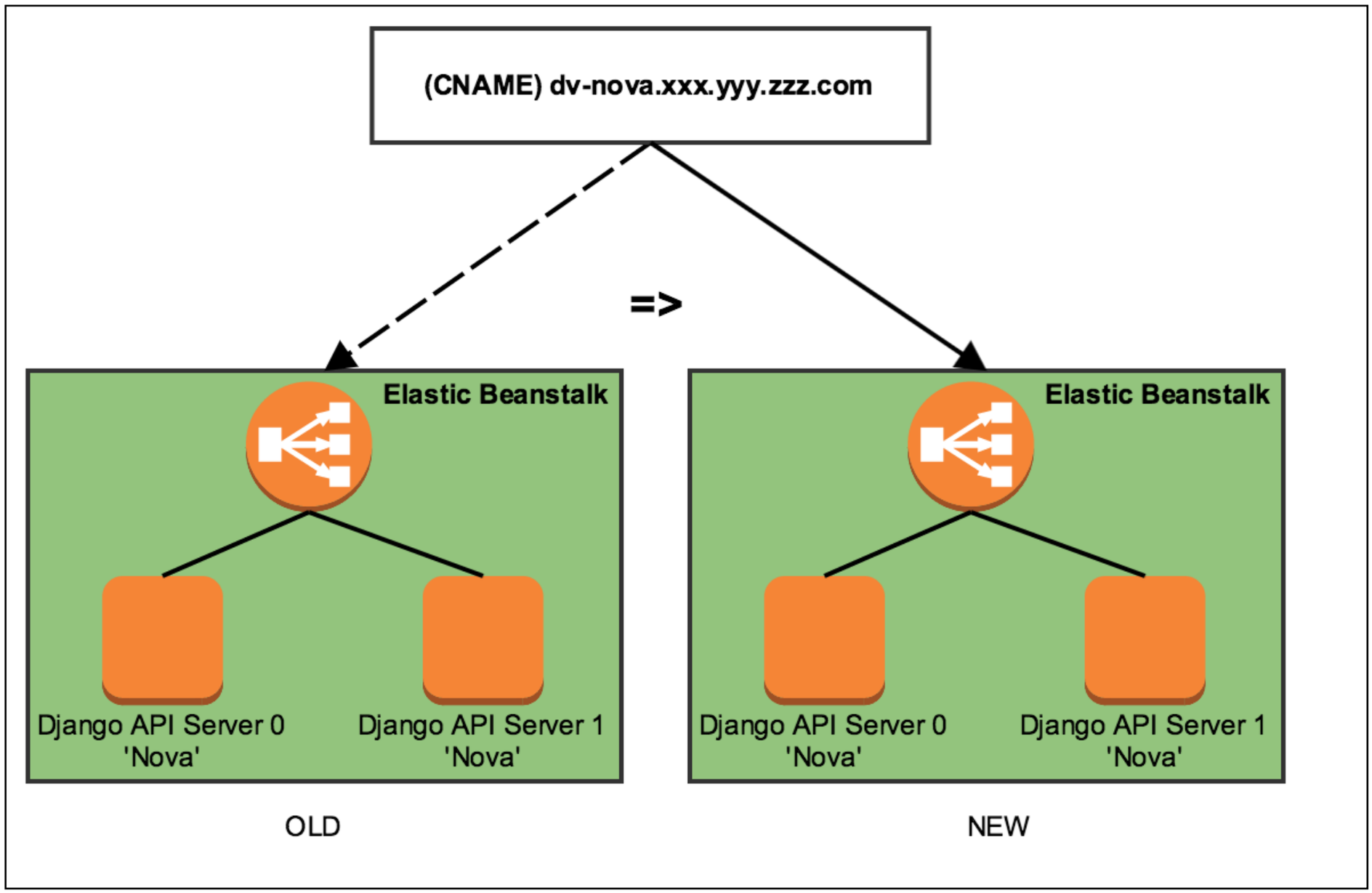


Live Demo 내용 소개 (cont.)

PYCON APAC 2016

Kanizsa Lab

- CNAME Swapping:
  -> Daily CD의 핵심
  -> 기존 'Nova'를 새로운 'Nova'로 교체 (no downtime)



Live Demo 내용 소개 (cont.)

PYCON APAC 2016

Kanizsa Lab

- CNAME Swapping (before)

All Applications

dv

nova-1470573984

Environment tier: Web Server

Running versions: app-160807_214634

Last modified: 2016-08-07 21:50:52 UTC+0900

URL: addnull-dv-nova.ap-northeast-2.elasticbeanstalk.com

Live Demo 내용 소개 (cont.)

PYCON APAC 2016

Kanizsa Lab

- CNAME Swapping (in progress)

All Applications

dv

nova-1470573984

Environment tier: Web Server

Running versions: app-160807_214634

Last modified: 2016-08-07 21:50:52 UTC+0900

URL: addnull-dv-nova.ap-northeast-2.elasticbeanstalk.com

nova-1470577190

Environment tier: Web Server

Running versions:

Last modified: 2016-08-07 22:40:13 UTC+0900

URL: addnull-dv-nova-1470577190.ap-northeast-2.elasticbeanstal…

Live Demo 내용 소개 (cont.)

PYCON APAC 2016

Kanizsa Lab

- CNAME Swapping (after)

All Applications

dv

nova-1470573984

Environment tier: Web Server

Running versions: app-160807_214634

Last modified: 2016-08-07 22:44:29 UTC+0900

URL: addnull-dv-nova-1470577190.ap-northeast-2.elasticbeanstal…

nova-1470577190

Environment tier: Web Server

Running versions: app-160807_224003

Last modified: 2016-08-07 22:44:29 UTC+0900

URL: addnull-dv-nova.ap-northeast-2.elasticbeanstalk.com

'Johanna' command

PYCON APAC 2016

Kanizsa Lab

- command set
  - create
  - create_vpc
  - create_nova
  - terminate
  - terminate_nova
  - terminate_vpc
  - terminate_eb_old_environment

'Johanna' command (cont.)

PYCON APAC 2016

Kanizsa Lab

- './run.py'

```
16-08-07 22:44:23 [addnull@null-mbp-os-x-11 johanna] 10020 0 ➜ (johanna|master) ./run.py
Your current environment values are below
------------------------------------------------------------------------
        PHASE             : 'dv'
        CNAME of Nova     : 'addnull-dv-nova'
------------------------------------------------------------------------
Please type in the name of phase 'dv' to confirm: dv
########################################################################
How to Play

------------------------------------------------------------------------
    ./run.py create
    ./run.py create_nova
    ./run.py create_vpc
    ./run.py terminate
    ./run.py terminate_eb_old_environment
    ./run.py terminate_nova
    ./run.py terminate_vpc
------------------------------------------------------------------------
    ./run.py [AWS CLI COMMAND]                (ex: './run.py -- aws ec2 describe-instances')
    cd [EB DIR]; ../run.py [EB CLI COMMAND]   (ex: 'cd nova; ../run.py -- eb list --region ap-northeast-2')

########################################################################
16-08-07 22:58:09 [addnull@null-mbp-os-x-11 johanna] 10021 0 ➜ (johanna|master) _
```

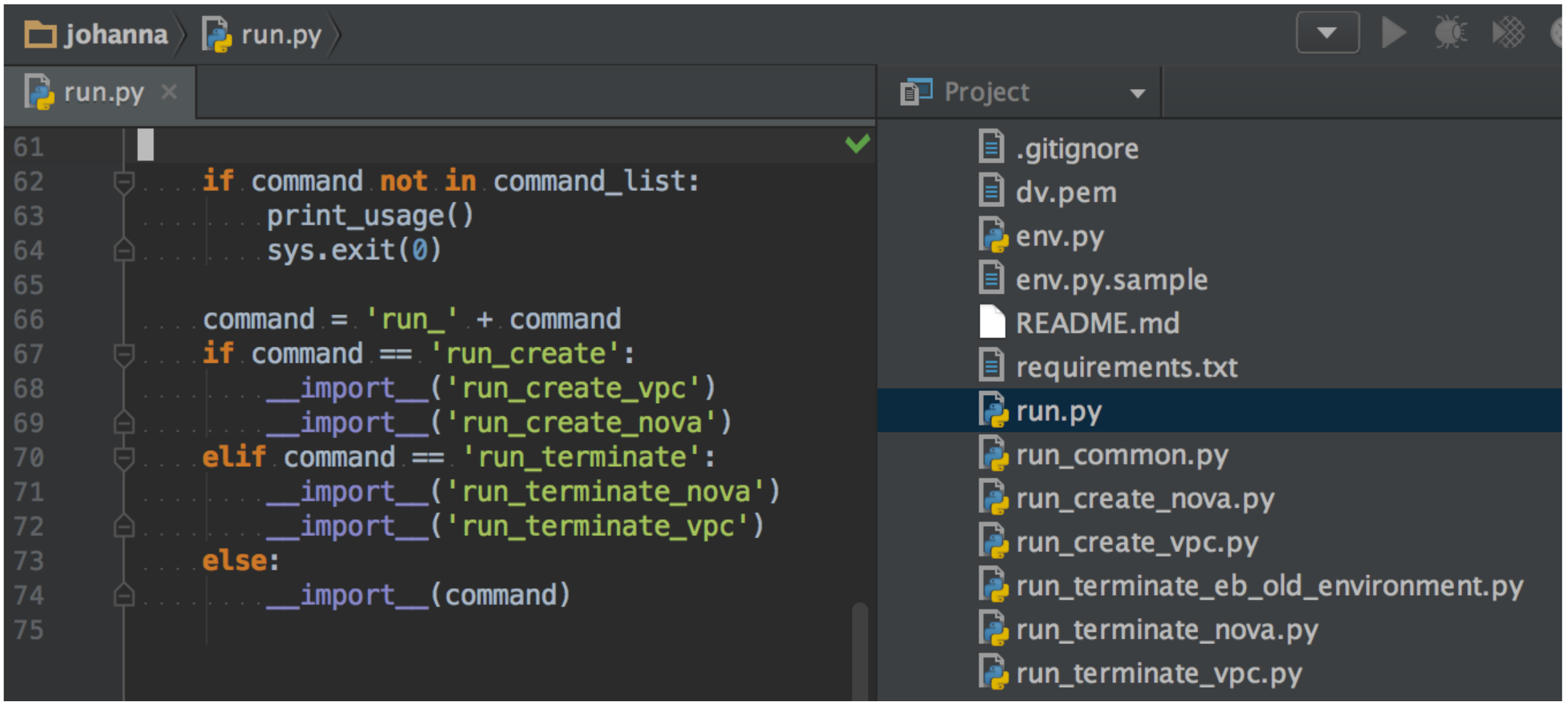
'Johanna' command (cont.)

PYCON APAC 2016

Kanizsa Lab

- 'create'
  -> 'create_vpc' 실행 후 'create_nova' 실행. (re)provisioning

- 'terminate'
  -> 'terminate_nova' 실행 후 'terminate_vpc' 실행. deprovisioning



'Johanna' command (cont.)

PYCON APAC 2016

Kanizsa Lab

- 'create_vpc'
  -> VPC 구성과 기타(IAM, ssh key) 설정 생성

- 'create_nova'
  -> 새로운 'nova'를 생성
  -> 기존에 'nova'가 존재한다면, CNAME swapping
  -> 즉, daily CD용 command

- 'terminate_nova'
  -> 'create_nova'의 반대

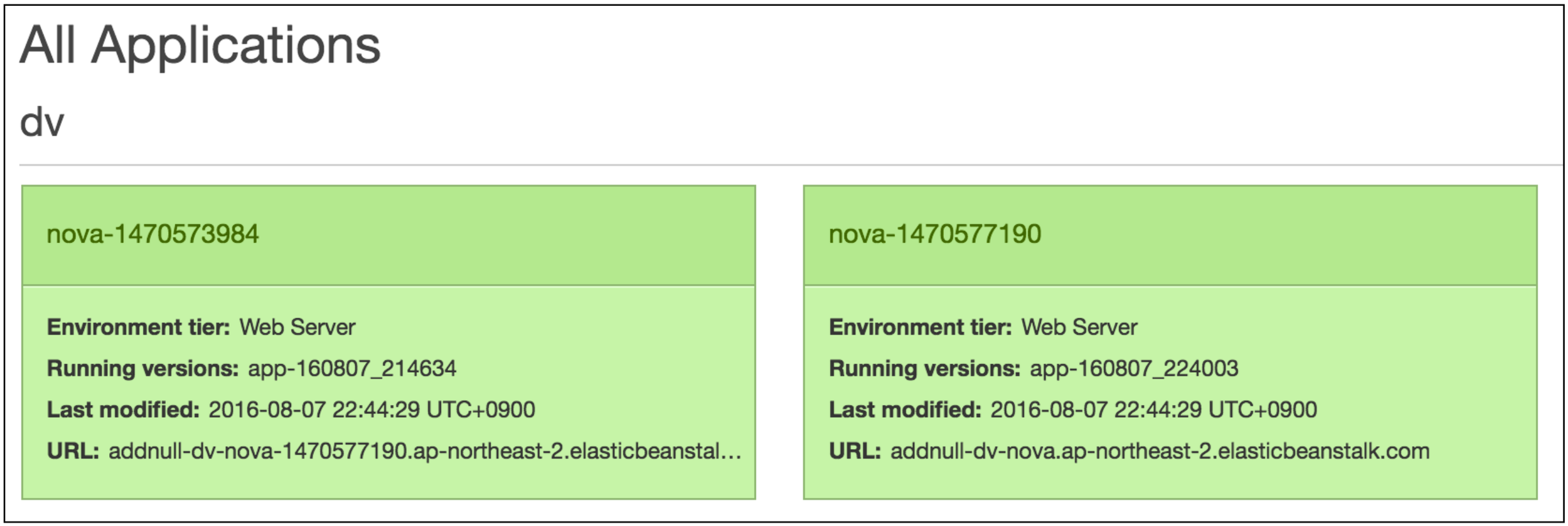
- 'terminate_vpc'
  -> 'create_vpc'의 반대

'Johanna' command (cont.)

PYCON APAC 2016

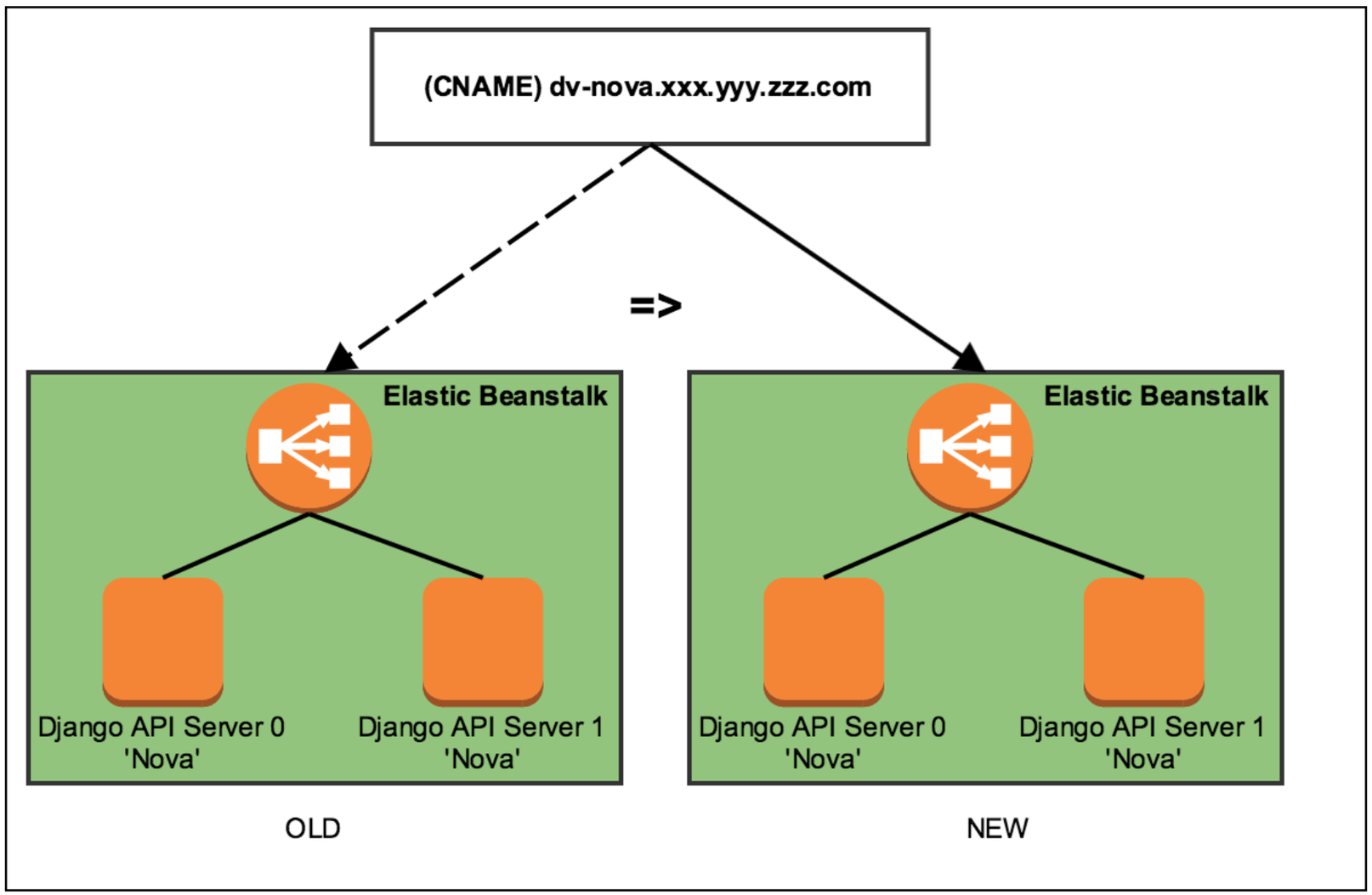
Kanizsa Lab

- terminate_eb_old_environment
  CNAME swap된 이전 Elastic Beanstalk environment를 제거
  (아래 이미지에서 왼쪽)



'Johanna' command (cont.)

- terminate_eb_old_environment
  왜 CNAME swap된 이전 Elastic Beanstalk environment를
  바로 제거하지 않는가?
  -> local routing table cache 문제



Custom CLI 개발

PYCON APAC 2016

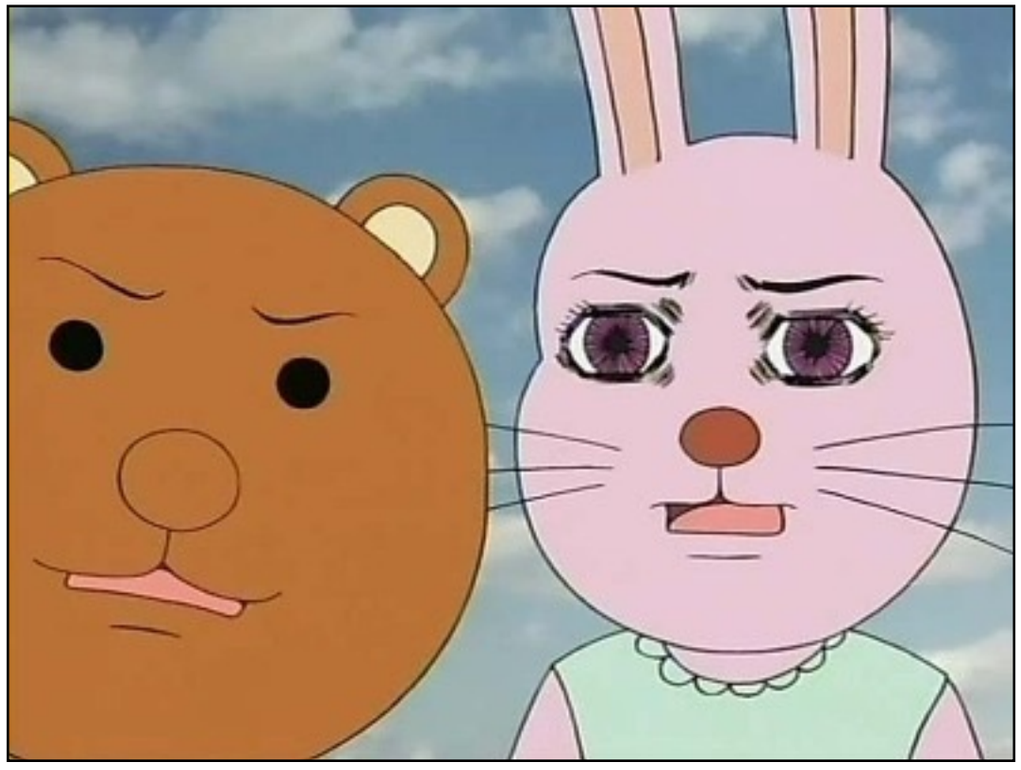
Kanizsa Lab

왜 Python인가?

PYCON APAC 2016

Kanizsa Lab

- 발표 자료 30장이 넘어서, 드디어 Python 이야기를 ...



왜 Python인가? (cont.)

PYCON APAC 2016

Kanizsa Lab

- 이전에 bash script로 했을때 문제점들
  => SED hell ...
  => JSON string handling

- Chef, Puppet, Ansible을 선택하지 않은 이유
  => simple is best

- Boto/Boto3 대신에 AWS/EB CLI를 사용한 이유
  => 'do not need to reinvent the wheel'

Custom CLI 만들기

- 기존의 AWS/EB CLI를 wrapping하는 'run.py'
- '--'를 이용해서 AWS/EB CLI와 'run.py'의 option을 구별
  (예: 아래 이미지에서 EB CLI의 '--region')
- AWS CLI 결과는 (대부분) JSON 이지만, EB CLI 결과는 multi-line string

```
./run.py [AWS CLI COMMAND]                (ex: './run.py -- aws ec2 describe-instances')
cd [EB DIR]; ../run.py [EB CLI COMMAND]   (ex: 'cd nova; ../run.py -- eb list --region ap-northeast-2')
```

AWS CLI =>

EB CLI =>

```python
if command == 'aws':
    aws_cli = AWSCli()
    result = aws_cli.run(args[2:], ignore_error=True)
    if type(result) == dict:
        print json.dumps(result, sort_keys=True, indent=4)
    else:
        print result
    sys.exit(0)
elif command == 'eb':
    aws_cli = AWSCli()
    result = aws_cli.run_eb(args[2:], ignore_error=True)
    print result
    sys.exit(0)
```

Custom CLI 만들기 (cont.)

PYCON APAC 2016

Kanizsa Lab

- 기존의 AWS/EB CLI를 wrapping하는 'run.py'

```
./run.py [AWS CLI COMMAND]                (ex: './run.py -- aws ec2 describe-instances')
cd [EB DIR]; ../run.py [EB CLI COMMAND]   (ex: 'cd nova; ../run.py -- eb list --region ap-northeast-2')
```

AWS CLI

```
##########################################
print_message('create vpc')

cmd = ['ec2', 'create-vpc']
cmd += ['--cidr-block', cidr_vpc['eb']]
result = aws_cli.run(cmd)
eb_vpc_id = result['Vpc']['VpcId']
aws_cli.set_name_tag(eb_vpc_id, 'eb')
```

EB CLI

```
##########################################################
print_message('create nova')

tags = list()
tags.append('git_hash_johanna=%s' % git_hash_johanna)
tags.append('git_hash_nova=%s' % git_hash_nova)

cmd = ['create', eb_environment_name]
cmd += ['--cname', cname]
cmd += ['--instance_type', 't2.nano']
cmd += ['--region', aws_default_region]
cmd += ['--tags', ','.join(tags)]
cmd += ['--vpc.ec2subnets', subnet_id_1 + ',' + subnet_id_2]
cmd += ['--vpc.elbpublic']
cmd += ['--vpc.elbsubnets', subnet_id_1 + ',' + subnet_id_2]
cmd += ['--vpc.id', eb_vpc_id]
cmd += ['--vpc.publicip']
cmd += ['--vpc.securitygroups', security_group_id]
cmd += ['--quiet']
aws_cli.run_eb(cmd, cwd='nova')
```

Custom CLI 만들기 (cont.)

PYCON APAC 2016

Kanizsa Lab

- 'subprocess.Popen'으로 AWS/EB CLI를 실행
- 'stdout'를 PIPE 설정하면, 결과 내용을 저장할 수 있음.
- 'cwd'를 이용해서 실행 path를 변경할 수 있음.

```
def run(self, args, cwd=None, ignore_error=None):
    args = ['aws'] + args
    return self._run(args, cwd, ignore_error)

def run_eb(self, args, cwd=None, ignore_error=None):
    args = ['eb'] + args
    return self._run(args, cwd, ignore_error)
```

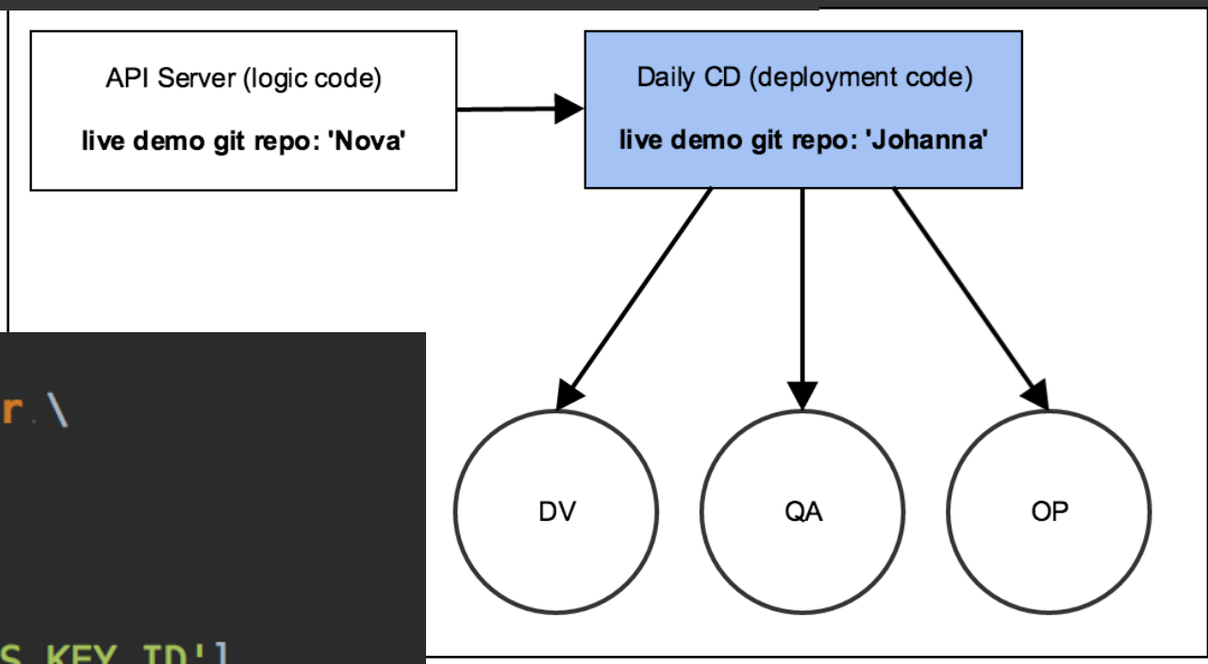
```
def _run(self, args, cwd=None, ignore_error=None):
    if ignore_error:
        print '\n>> command(ignore error):',
    else:
        print '\n>> command:',
    print ' '.join(args)
    result, error = subprocess.Popen(args, stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                                     cwd=cwd, env=self.env).communicate()
```

Custom CLI 만들기 (cont.)

PYCON APAC 2016

Kanizsa Lab

- AWS/EB CLI 를 위한 'env' 설정




```
if not env['aws'].get('AWS_ACCESS_KEY_ID') or \
        not env['aws'].get('AWS_SECRET_ACCESS_KEY') or \
        not env['aws'].get('AWS_DEFAULT_REGION'):
    raise Exception()

self.env = dict(os.environ)
self.env['AWS_ACCESS_KEY_ID'] = env['aws']['AWS_ACCESS_KEY_ID']
self.env['AWS_SECRET_ACCESS_KEY'] = env['aws']['AWS_SECRET_ACCESS_KEY']
self.env['AWS_DEFAULT_REGION'] = env['aws']['AWS_DEFAULT_REGION']
```

```
def _run(self, args, cwd=None, ignore_error=None):
    if ignore_error:
        print '\n>> command(ignore error):',
    else:
        print '\n>> command:',
    print ' '.join(args)
    result, error = subprocess.Popen(args, stdout=subprocess.PIPE, stderr=subprocess.PIPE,
                                     cwd=cwd, env=self.env).communicate()
```

Custom CLI 만들기 (cont.)

PYCON APAC 2016

Kanizsa Lab

- 'subprocess.call'을 사용하지 않은 이유

> **Note:** Do not use `stdout=PIPE` or `stderr=PIPE` with this function as that can deadlock based on the child process output volume. Use **`Popen`** with the **`communicate()`** method when you need pipes.

Custom CLI 만들기 (cont.)

PYCON APAC 2016

Kanizsa Lab

- '-f', '--force' option 지원 -> OptionParser를 이용

```
./run.py -f -- aws ec2 describe-instances
```

```python
if __name__ == "__main__":
    from run_common import parse_args

    args = parse_args(True)
```

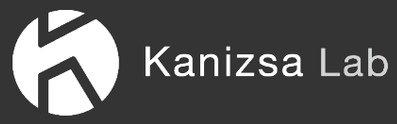
```python
def parse_args(require_arg=False):
    if require_arg:
        usage = 'usage: %prog [options] arg'
    else:
        usage = 'usage: %prog [options]'

    parser = OptionParser(usage=usage)
    parser.add_option("-f", "--force", action="store_true", help='skip the phase confirm')
    (options, args) = parser.parse_args(sys.argv)

    if not options.force:
        _confirm_phase()

    return args
```

지난 6달간 실제 서비스로 운영 경험



난 할 수 있다

- 많은 고려 사항들. 잊을 만하면 발생되는 버그.

(출처: 한겨레)

Lesson Learned

PYCON APAC 2016

Kanizsa Lab

- 장점
  - 패치 작업의 일상화로 인한 심적 부담감이 낮아지고, fail-over가 빠르고 쉬워짐.
  - memory leak 문제를 신경쓰지 않음.
  - 오랜 기간 서버를 운영할 시에 발생되는 장애가 없어짐 (예: 한달에 한번 발생되는 비정상 동작)
  - 새로 입사한 개발자에게 바로 개발 환경을 제공 가능

Lesson Learned (cont.)

- 장점
  - 별도의 QA 환경을 원하는 때에만 운영이 가능
    -> 즉, 비용 절감
  - AWS EC2의 최저 사양 t2.nano만으로도 전체 backend infra 구축이 가능해짐.
    -> 매일 새로 EC2를 생성하므로 추가 CPU credit을 얻는 것 같은 효과가 발생

Lesson Learned (cont.)

PYCON APAC 2016

Kanizsa Lab

- 단점
  - AWS 환경, Linux package repository 변화에 따른 예상치 못한 장애 (즉, 가장 빠른 beta tester로서 겪는 문제점)
  - Daily CD python script에 대한 유지보수
    -> 수정 빈도가 낮은 편이지만, 이전에 왜 이렇게 했는지 의도를 파악하기 어려움

Wrap Up

PYCON APAC 2016

Kanizsa Lab

발표 내용 Wrap Up

- 'Daily Continuous Deployment' (이하 'daily CD') 소개 및 장단점
- 요구 사항들
- GitHub repo 'Nova', 'Johanna' 소개 및 Live Demo
- Custom CLI 개발
- 지난 6달간 실제 서비스로 운영 경험
- Wrap Up & QnA

Q n A

한종원
addnull@gmail.com
010-9166-6855

PYCON APAC 2016

Kanizsa Lab