Session Mirroring Test ENV 구성 방안 제안

Customer Pain Point

- Oracle의 SP(Stored Procedure)를 분리하기 위한 별도의 Test 환경이 없기 때문에 분리후 검증 절차에 어려움을 느낌
- SP 분리 방식의 경험이 부족하여 체계적인 방법이 필요함

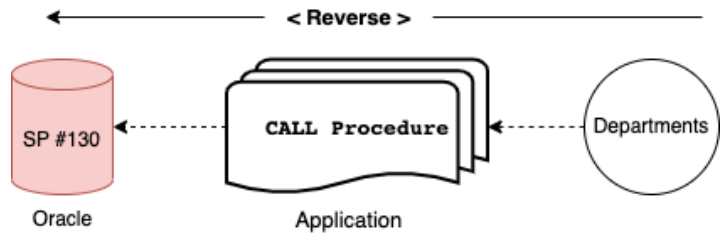
SP Decompose Strategy

`TVING의 Oracle SP(Stored Procedure)를 보다 효율적으로 Decompose하여 Application 로직으로 변환하기 위한 절차를 타사 사례를 통해 제안함.(TVING과 같은 구조의 SP를 운영하며 분리를 위한 프로세스를 확립하고 보다 효율적으로 나누어 로직을 분리함)

PROGRESS

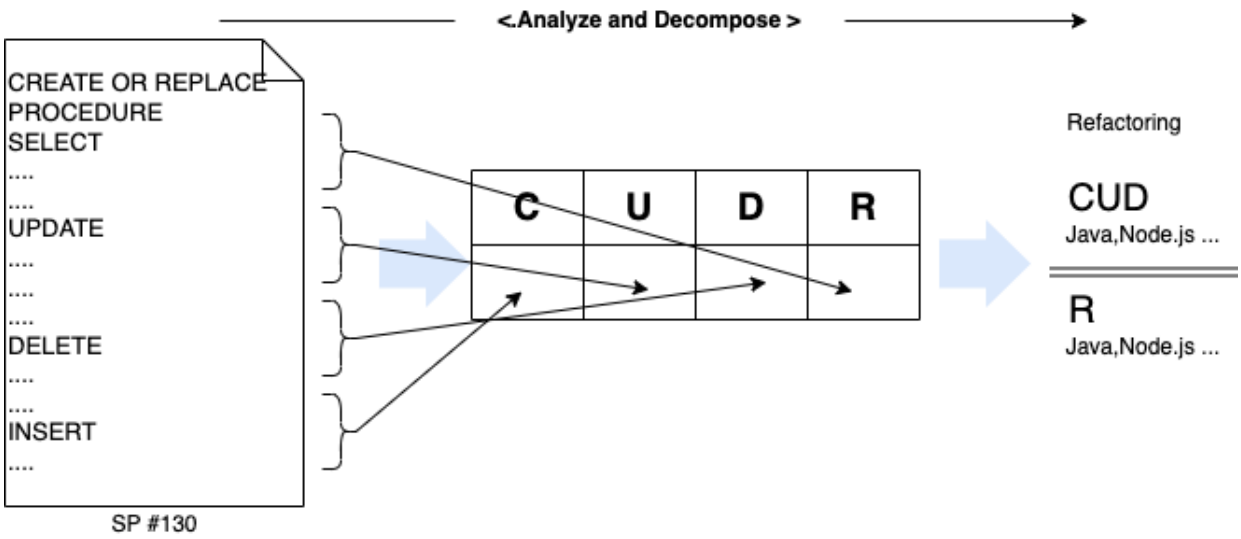
1. Identify Target

사전 미사용 SP에 대한 선별 작업은 Procedure Call Log 분석을 통해 제거후 대상 SP를 축소합니다. 이 작업은 SP를 선 분석하기 보다 역 방향으로 사전에 미사용 또는 unknown SP를 식별하기 위함입니다.

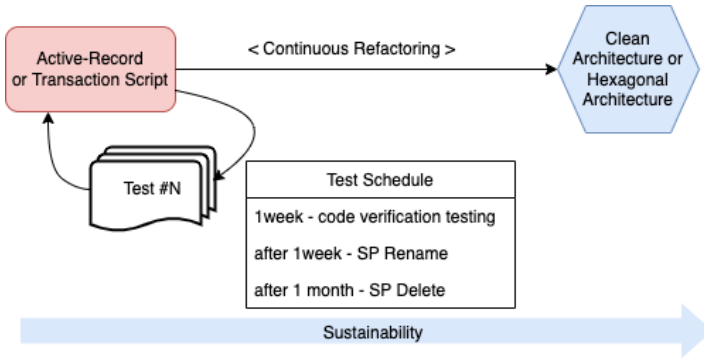


Procedure call log를 분석하여 대상을 식별합니다. 이때 각 부서별 Application 담당자가 담당하고 있는 Application별 분석을 주도합니다. SP에 존재하는 Business Logic의 History가 있다면 참조하여 아래의 절차를 진행 해도 무관합니다.

2. Analyze and Decompose



대상이 식별되면 해당 SP의 CURD별로 분리하여 각 담당자가 해당 SP에 있는 Business Logic을 Application Code로 Converting 합니다. 이 Application은 아래 Test Environment를 활용하여 정상 작동 여부를 판단할 수 있습니다.



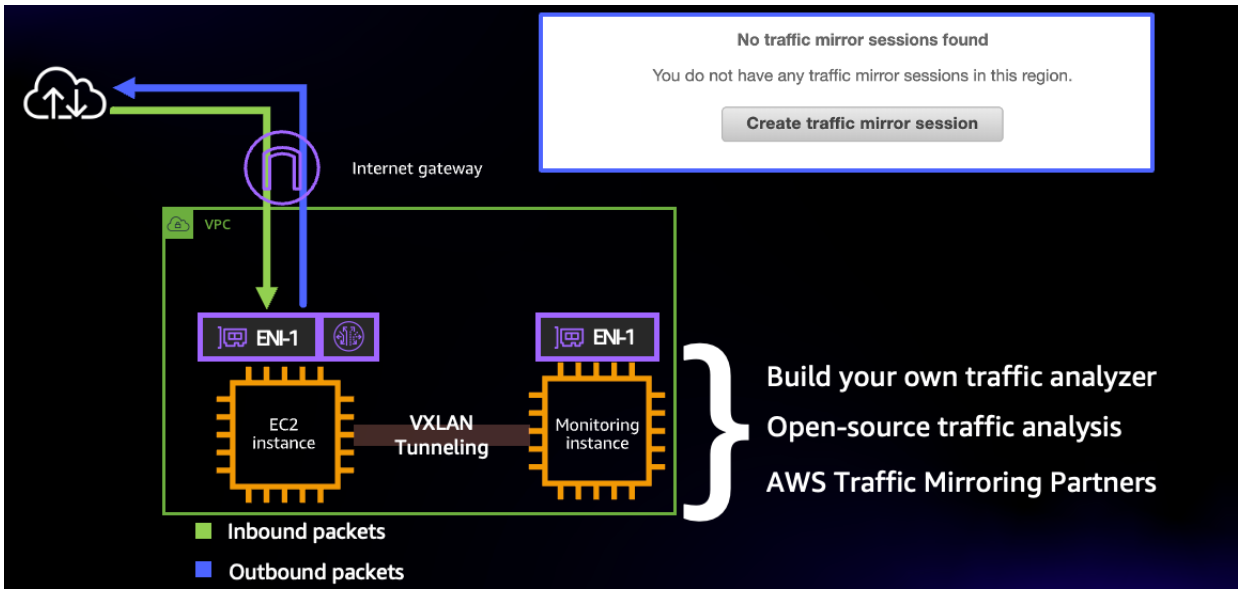
우선은 SP를 Converting한 코드가 정상 동작 여부를 확인합니다. 정상적으로 동작하게 되면 해당 SP의 Name을 변경하여 중첩된 다른 SP의 호출 여부를 확인하며 side effect를 확인합니다.
중첩 호출과 코드의 정상 동작을 확인했다면 Refactoring을 진행하며 지속적으로 Inner Architecture를 개선합니다.

> 테스트는 최대 1주일까지 지켜보며 1주일후 기존 SP를 Rename하고 1개월동안 확인합니다. 이는 중첩된 호출에 대한 이슈를 사전에 파악하여 side effect를 최소화 합니다.

정상 동적 여부를 확인한뒤 Converting된 Business Logic과 Package 구조는 지속적으로 refactoring하여 code smell을 제거해 나갑니다.

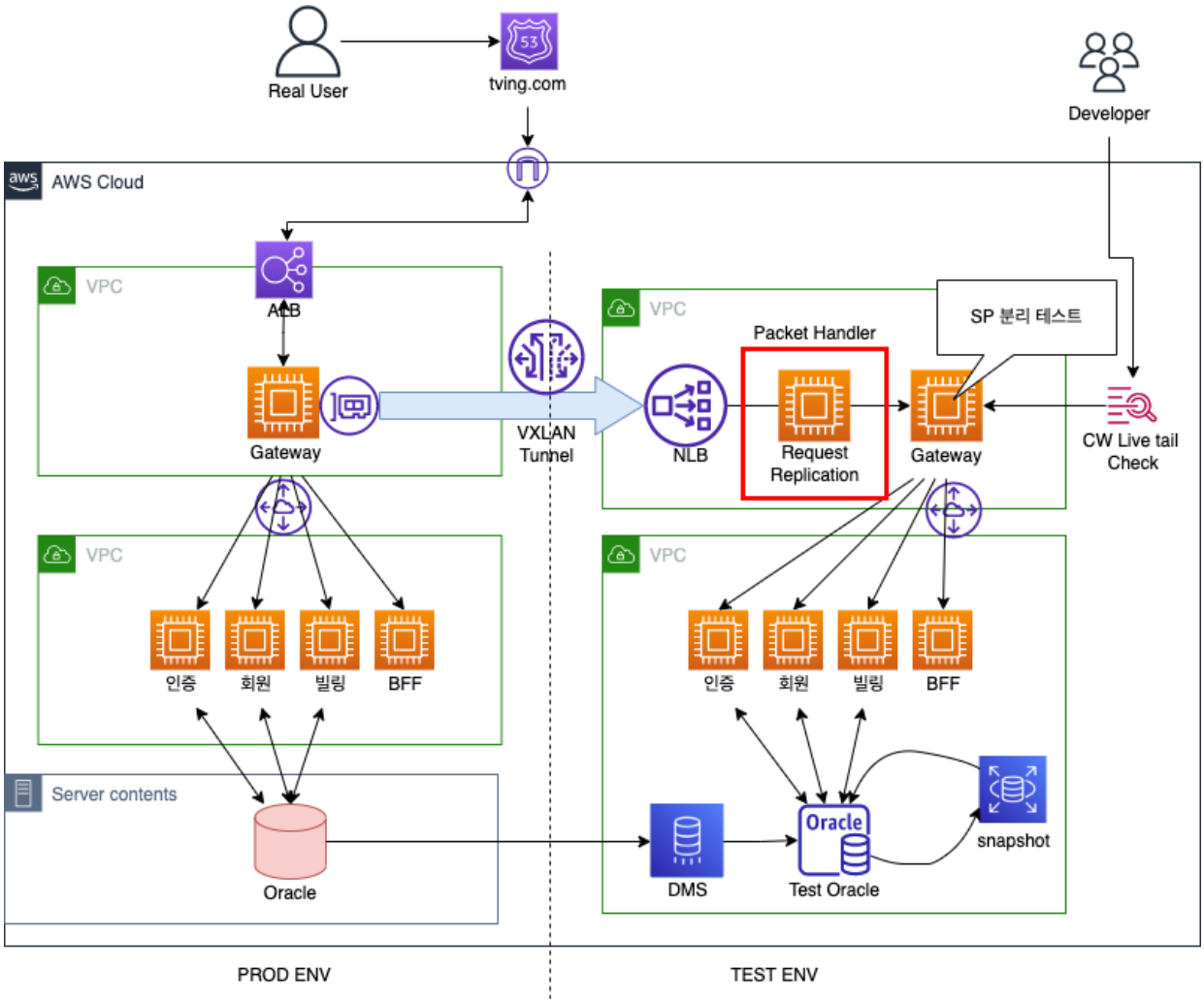
Test Environment Objective

TVING의 Oracle SP(Stored Procedure)의 비즈니스 로직을 분리하여 Application 내의 Business Domain으로 아래와 같이 VPC의 Session Mirroring 기능을 활용한 개발 환경을 제안함



> VPC Traffic Mirroring은 VXLAN Tunneling을 이용하여 Source ENI의 Traffic을 L4 Layer에서 복제하여 대상에 전달합니다.(대상은 ENI, NLB, GWLB로 지정이 가능합니다)

제안 ARCHITECTURE



Session Mirroring은 VXLAN을 통한 전송이기 때문에 Encapsulation 되어 있어 Decoding 할 수 있는 Filter가 탑재된 솔루션들만 Decoding할 수 있습니다.(이 기능은 Nginx와 Envoy가 제공함)

- http://nginx.org/en/docs/http/ngx_http_mirror_module.html
- https://www.envoyproxy.io/docs/envoy/latest/api-v3/config/route/v3/route_components.proto.html#envoy-v3-api-msg-config-route-v3-routeaction-requestmirrorpolicy

아래와 같이 GO를 이용하여 개발된 별도의 Proxy Web Server를 활용할 수 있습니다.

- https://github.com/aws-samples/http-requests-mirroring/blob/main/main.go

EC2 Userdata

```
#!/bin/bash -xe
### update and install ###
export HOME=~
yum update -y
yum install git -y
yum install go -y
# dependency of github.com/google/gopacket
yum install install libpcap-devel -y
```

```
### dependency of main.go ###
go env GOPATH
go env -w GO111MODULE=off
echo 'export GOPATH=$HOME/go' >>~/.bash_profile
source ~/.bash_profile
go get "github.com/google/gopacket"
### create a virtual network interface that gets decapsulated VXLAN packets
#### compile & run go script ###
mkdir $GOPATH"/src/vxlan-to-http-request"
wget https://github.com/aws-samples/http-requests-mirroring/raw/main/main.go -P $GOPAT
go install "vxlan-to-http-request"
sudo ip link add vxlan0 type vxlan id $VNI dev eth0 dstport 4789
sudo ip link set vxlan0 up
$GOPATH"/bin/vxlan-to-http-request" -destination $TARGET -percentage "100" -percentage-
```

Overlay를 사용하고 UDP 4789를 사용하기 때문에 Packet Handler의 SG에 해당 inbound port가 open 되어 있고 auto scaling과 같은 구성을 취한다면 보다 높은 Traffic또한 Target으로 보낼수 있습니다.

기존 운영 환경의 축소한 환경을 별도의 VPC에 구성하고 RDS for Oracle에 동일한 데이터 및 기존 SP를 Migration 합니다. Traffic Mirroring을 통해 Gateway에서 들어오는 부하중 일부를 전달하여 지속적으로 상태를 갱신하며 변경 대상을 제거하고 Application을 개선해 Test환경에 배포후 정상 동작 여부를 확인합니다.

REFERENCE

- https://aws.amazon.com/ko/blogs/networking-and-content-delivery/mirror-production-traffic-to-test-environment-with-vpc-traffic-mirroring/
- https://github.com/aws-samples/http-requests-mirroring