**Welcome to Unicorn Rentals!**

**Background:**

For our next-generation application, we are taking a “Microservices Mesh” approach.  
Every team has a copy of the frontend API (called the "Service Router"). The Service Router uses microservices from other internal teams.

Your team is running one of the many test-in-production Service Routers. It receives requests containing a data string, and a list of operations to be completed before it responds.  
The Service Router forwards the data string to the required microservice for each operation.  
The Service Router will find these microservice endpoints by looking in a DynamoDB table called “service-table”.

차세대 애플리케이션을 위해 우리는 "마이크로 서비스 메시"접근 방식을 취하고 있습니다.

모든 팀에는 프런트 엔드 API ( "서비스 라우터"라고 함) 사본이 있습니다. 서비스 라우터는 다른 내부 팀의 마이크로 서비스를 사용합니다.

팀은 많은 프로덕션 테스트 서비스 라우터 중 하나를 실행하고 있습니다. 데이터 문자열이 포함 된 요청과 응답하기 전에 완료 할 작업 목록을 수신합니다.

서비스 라우터는 각 작업에 필요한 마이크로 서비스에 데이터 문자열을 전달합니다.

서비스 라우터는 "service-table"이라는 DynamoDB 테이블을 살펴봄으로써 이러한 마이크로 서비스 엔드 포인트를 찾습니다.

**High-Level Game Plan:**

* Deploy, Maintain and Optimize microservices for other teams to use.
* Publish your services using the Dashboard inputs. This publishes your microservice on the "Services Marketplace"
* Make your Service Router highly available. It comes attached to an Application Load Balancer and Autoscaling group which could be configured for HA.
* Use the best microservices yourself (put them in your Dynamo table). Discover them via the Services Marketplace link in the dashboard.
* Don't consume your own services!
* Keep everything going (use X-Ray, CW Metrics, CW Logs). Watch your [score events](https://dashboard.eventengine.run/score-events) for changes.
* 다른 팀이 사용할 수 있도록 마이크로 서비스를 배포, 유지 관리 및 최적화합니다.
* 대시 보드 입력을 사용하여 서비스를 게시합니다. 그러면 "서비스 마켓 플레이스"에 마이크로 서비스가 게시됩니다.
* 서비스 라우터의 가용성을 높입니다. HA 용으로 구성 할 수있는 Application Load Balancer 및 Autoscaling 그룹에 연결됩니다.
* 최고의 마이크로 서비스를 직접 사용합니다 (Dynamo 테이블에 배치). 대시 보드의 Services Marketplace 링크를 통해 찾아보십시오.
* 자신의 서비스를 사용하지 마십시오!
* 모든 것을 계속 진행합니다 (X-Ray, CW 메트릭, CW 로그 사용). 점수 이벤트에서 변경 사항을 확인하십시오.

**The Microservices:**

To maximise your score, you need to run copies of the microservices.  
When another teams Service Router receives a request, they may use your microservice to perform some of the required operations. If so, you get a share of their score.

점수를 극대화하려면 마이크로 서비스의 복사본을 실행해야합니다.  
다른 팀 서비스 라우터가 요청을 받으면 마이크로 서비스를 사용하여 필요한 작업 중 일부를 수행 할 수 있습니다. 그렇다면 점수의 일부를 얻습니다.

| **Service Name** | **Type** | **Instructions** |
| --- | --- | --- |
| swapcaser | SAM(Lambda and API Gateway) | [**Service and Instructions**](https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018-loadgen/v4/swapcaser.zip) |
| reverser | Fargate | [**Instructions**](https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018/assets/reverser-readme.txt) |
| leeter | Elastic Beanstalk | [**Service and Instructions**](https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018-loadgen/v2/leeterv2.zip) |

**How do I know if my Microservice is working?  
우리팀의 마이크로 서비스가 작동하는지 어떻게 알 수 있습니까?**  
First, try using the testsvc program provided [in this zip file](https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018/assets/testsvc.zip). This will test the correctness of your microservice and provide some information about what is wrong if it is not working properly. testsvc is also useful to test other team's microservices. There is a README with usage information in the zip file.

먼저 zip 파일에 제공된 testsvc 프로그램을 사용해보십시오. 그러면 마이크로 서비스의 정확성을 테스트하고 제대로 작동하지 않는 경우 무엇이 잘못되었는지에 대한 정보를 제공합니다. testsvc는 다른 팀의 마이크로 서비스를 테스트하는데도 유용합니다. zip 파일에 사용 정보가있는 README가 있습니다.

Next, add the service's endpoint/URL into the correct input in the Dashboard. The Service Market place will report its health. If you see latency and success rate of 0, but other team's services have values, you should re-check your microservice.

다음으로 서비스의 엔드 포인트 / URL을 대시 보드의 올바른 입력에 추가하십시오. 서비스 마켓 플레이스는 서비스상태(헬스체크) 알려드립니다. 지연 시간과 성공률이 0이지만 다른 팀의 서비스에 값이있는 경우 마이크로 서비스를 다시 확인해야합니다.

**How to simulate a real request to my Microservice?**   
**우리팀의 마이크로 서비스에 대한 실제 요청을 시뮬레이션하는 방법은 무엇입니까?**  
You can simulate a real user by using the testsvc program provided [in this zip file](https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018/assets/testsvc.zip). You could also use cURL, Httpie or Postman, etc. For example:

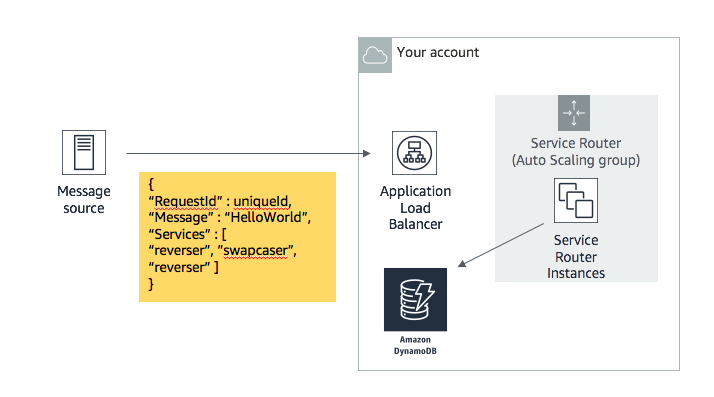
이 zip 파일에 제공된 testsvc 프로그램을 사용하여 실제 사용자를 시뮬레이션 할 수 있습니다. cURL, Httpie 또는 Postman 등을 사용할 수도 있습니다. 예를 들면 다음과 같습니다.

curl -d '{"Message":"TestMessage"}' -H 'Content-Type: application/json' URI

#Or with HTTPie

http POST URI Message='TestMessage'

**The Service Router:**

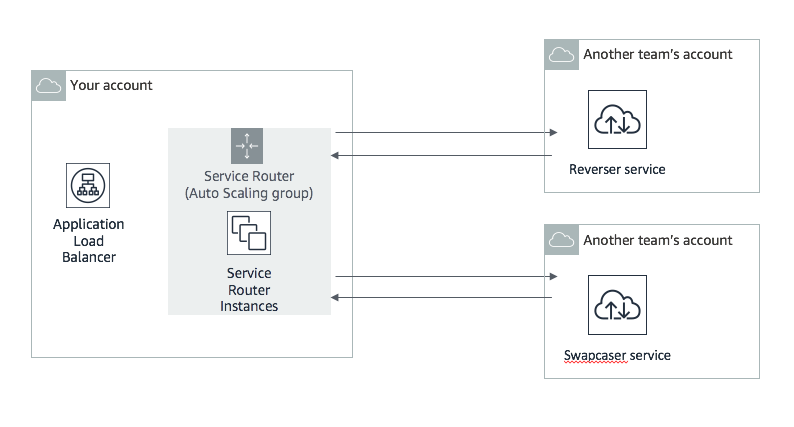
**Your Services Router**  


**The Services Router looks up which microservice to send messages to, using the DynamoDB “service-table" table**

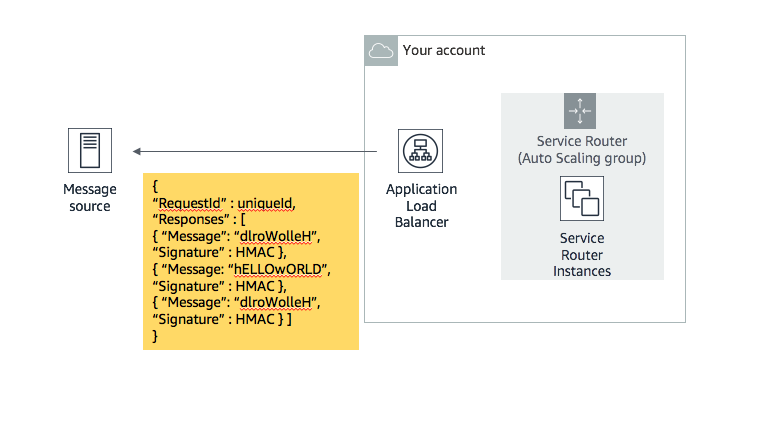
서비스 라우터는 DynamoDB "service-table"테이블을 사용하여 메시지를 보낼 마이크로 서비스를 조회합니다.

Edit the DynamoDB table to update the microservices your Service Router uses. You want to use many services, but also only the fastest ones. If you use slower services, you get less score. The Legacy Services provided at the start of the game are very slow.

DynamoDB 테이블을 편집하여 서비스 라우터가 사용하는 마이크로 서비스를 업데이트합니다. 많은 서비스를 사용하고 싶지만 가장 빠른 서비스 만 사용하려고합니다. 느린 서비스를 사용하면 점수가 낮아집니다. 게임 시작시 제공되는 레거시 서비스는 매우 느립니다.

**The Services Router then calls the relevant microservice endpoint in another DevOps team's account:**  
그런 다음 서비스 라우터는 다른 DevOps 팀의 계정에서 관련 마이크로 서비스 엔드 포인트를 호출합니다.

**Once it has received the responses from the other team(s), the Service Router returns the correct payload to the sender  
다른 팀으로부터 응답을 받으면 서비스 라우터는 올바른 페이로드를 보낸 사람에게 반환합니다.**  
The messages contain signatures from other teams, proving they did some work. Your team scores points, and so do the other teams who did some work for you.

메시지에는 다른 팀의 서명이 포함되어 작업을 수행했음을 증명합니다. 당신의 팀은 점수를 얻고 당신을 위해 어떤 일을 한 다른 팀도 마찬가지입니다.  


**How does the Service Router know where where to send messages (aka Service Discovery)?** **서비스 라우터는 메시지를 보낼 위치 (서비스 검색)를 어떻게 알 수 있습니까?**

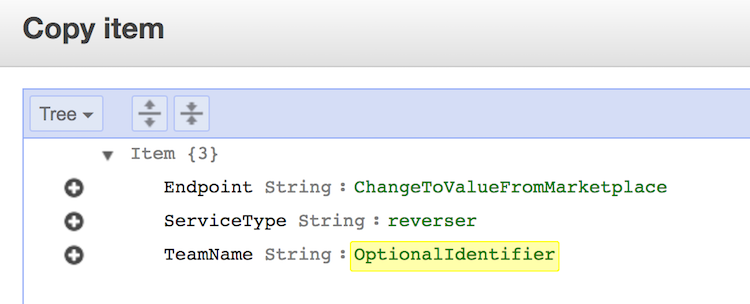
Your account comes with a “service-table" DynamoDB table pre-made. Four Legacy Services are already listed in there. However these were created by the last team (before they were fired), and are very slow.

계정에는 "서비스 테이블"DynamoDB 테이블이 미리 만들어져 있습니다. 4 개의 레거시 서비스가 이미 여기에 나열되어 있습니다. 그러나 이러한 서비스는 마지막 팀 (해고되기 전)에서 생성했으며 매우 느립니다.

| **DDB Field** | **Description** | **Required** |
| --- | --- | --- |
| Endpoint | URI of a microservice our service-router can use | Yes |
| ServiceType | Type of service (reverser, swapcaser, leeter) | Yes |
| TeamName | A field to help recall which teams service it is | No |

You will need to add new services to this table, and remove the old slow ones.  
An easy way to do this, is duplicate an old entry, and change the Endpoint and TeamName fields.

이 테이블에 새 서비스를 추가하고 이전의 느린 서비스를 제거해야합니다.

이를 수행하는 쉬운 방법은 이전 항목을 복제하고 Endpoint 및 TeamName 필드를 변경하는 것입니다.

**Building Service Routers 서비스 라우터 구축**  
Source is here:  
<https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018/assets/service-router.zip>

To build/rebuild your Service Router, use a userdata like below with Amazon Linux 2. Your account comes with an Autoscaling group already built this way.

서비스 라우터를 구축 / 재 구축하려면 Amazon Linux 2에서 아래와 같은 사용자 데이터를 사용하십시오. 계정에는 이미 이러한 방식으로 구축 된 Autoscaling 그룹이 함께 제공됩니다.

#!/bin/bash

export AWS\_DEFAULT\_REGION=<region\_the\_game\_is\_running\_in>

aws configure set region $AWS\_DEFAULT\_REGION

cd /home/ec2-user/

yum install -y unzip python-pip

curl -o xray.zip https://s3.amazonaws.com/aws-xray-assets.us-east-1/xray-daemon/aws-xray-daemon-linux-3.x.zip

#for ARM / a1 instances, use 'curl -o xray.zip https://s3.amazonaws.com/ee-assets-prod-us-east-1/modules/gd2018/assets/xray-arm.zip'

unzip xray.zip

chmod +x ./xray && nohup ./xray > xray.out 2>&1 &

curl -o service-router.zip https://ee-assets-prod-us-east-1.s3.amazonaws.com/modules/gd2018/assets/service-router.zip

unzip service-router.zip

cd service-router

pip install -r requirements.txt

python service-router.py

Learn about how userdata works  
사용자 데이터 작동 방식에 대해 알아보기   
here: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html>  
You can also login to the instance using Session  
세션을 사용하여 인스턴스에 로그인 할 수도 있습니다.  
Manager: <https://console.aws.amazon.com/systems-manager/session-manager/>

**Size Restrictions** The Service Router doesn't need to be big: a1.medium or t2 or t3 nano/micro/small

**크기 제한** 서비스 라우터가 클 필요는 없습니다 : a1.medium 또는 t2 또는 t3 nano / micro / small

**Operations FAQ:**

**How do I know if my Service Router is working?** **내 서비스 라우터가 작동하는지 어떻게 알 수 있습니까?**  
Test its endpoint in a browser. You should see the "service map" - a list of the services it has pulled from the DynamoDB table. For example:

브라우저에서 엔드 포인트를 테스트하십시오. DynamoDB 테이블에서 가져온 서비스 목록 인 "서비스 맵"이 표시되어야합니다. 예를 들면 :

Service Map: {'reverser': ['[http://reverser-url']](http://reverser-url%27]/), 'givemeasign': ['[http://givemeasign-url']](http://givemeasign-url%27]/), 'swapcaser': ['[http://swapcaser-url']}](http://swapcaser-url%27]%7D/)

**How do I know what my Service Router is doing?** **내 서비스 라우터가 무엇을하는지 어떻게 알 수 있습니까?**  
The Service Router is integrated with AWS X-Ray and Amazon CloudWatch Logs. If the instance/container/lambda has the 'PlayerInstanceProfile' or 'TEAMROLE' IAM role attached, it will send metrics and logs to those services.  
Cloudwatch Insights can be used to search the Service Routers logs for errors e.g:

서비스 라우터는 AWS X-Ray 및 Amazon CloudWatch Logs와 통합됩니다. 인스턴스 / 컨테이너 / 람다에 'PlayerInstanceProfile'또는 'TEAMROLE'IAM 역할이 연결되어 있으면 해당 서비스에 지표와 로그를 보냅니다.

Cloudwatch Insights를 사용하여 서비스 라우터 로그에서 오류를 검색 할 수 있습니다.

fields @timestamp, status, message

| filter status not like /info/

| sort @timestamp desc

| limit 20

**How do I run/test Service Router interactively? 서비스 라우터를 대화 형으로 실행 / 테스트하려면 어떻게합니까?**  
Use Session Manager (part of Systems Manager) to create a session to your instance. You will need to update the instances SSM Agent to do this. Follow the instructions in Session Manager console.

세션 관리자 (Systems Manager의 일부)를 사용하여 인스턴스에 대한 세션을 생성합니다. 이를 수행하려면 인스턴스 SSM 에이전트를 업데이트해야합니다. 세션 관리자 콘솔의 지침을 따르십시오.

sudo su ec2-user

cd /home/ec2-user/

cd service-router/

sudo python service-router.py

You may also want to get/start xray, to get request traces.  
요청 추적을 얻기 위해 xray를 가져 오거나 시작할 수도 있습니다.

curl -O https://s3.amazonaws.com/aws-xray-assets.us-east-1/xray-daemon/aws-xray-daemon-linux-2.x.zip

unzip aws-xray-daemon-linux-2.x.zip

nohup ./xray &

**How to simulate a real request to my Service Router?** **내 서비스 라우터에 대한 실제 요청을 시뮬레이션하는 방법은 무엇입니까?**  
You can simulate a real user by using cURL, Httpie or Postman (GUI API testing tool).  
cURL, Httpie 또는 Postman (GUI API 테스트 도구)을 사용하여 실제 사용자를 시뮬레이션 할 수 있습니다.

#curl https://curl.haxx.se/download.html:

curl -d '{"Message":"TestMessage","RequestId":"test","Services":["reverser","swapcaser"]}' -H 'Content-Type: application/json' URI

#curl (Windows escaping)

curl -d "{\"Message\":\"TestMessage\",\"RequestId\":\"test\",\"Services\":[\"reverser\",\"swapcaser\"]}" -H "Content-Type: application/json" URI

#Httpie https://httpie.org/

http POST URI Message='TestMessage' RequestId='test' Services:='["reverser","swapcaser"]'