



Full Stack Game Demo

Serverless Full Stack with Gomoku

Sep. 2019

Table of Contents

Overview of Game.....	4
Architecture Diagram	5
Let's start to stack the full stack!	5
Section 1: DynamoDB and ElastiCache – Redis and SQS. With a dash of IAM roles.	6
Section 2: Lambda Lambda Lambda	24
Section 3: Face the users with API Gateway and S3	32
Section 4: Putting all together in GameLift	42
Section 5: Serverless FlexMatch 구성하기	47
Section 6: Let's play the client.....	58
Appendix A Deployment Package with Python cheat sheet	60
Appendix B Notes on compiling the source binary.....	61
Appendix C AWS Cli environment notes	62
Appendix D Setting up Windows notes.....	63

HoL Material by

Version 1.0 Junghun Lee, Sep. 2019

Based on material prepared by Seungmo Koo, Sungsoo Khim, Hyobin An

Hands-on-Lab 실습용 Asset 다운로드

<https://leejungh-public.s3.ap-northeast-2.amazonaws.com/hol/GameLiftFlexHol.zip>

Source Code

<https://github.com/aws-samples/aws-gamelift-sample/tree/FlexMatch>

Overview of Game

지난 수개월간 만들어오던 여러분의 새로운 게임이 드디어 완성을 눈 앞에 두고 있습니다. 가로 세로 19 칸으로 그어진 선 위에서 우주만물의 음과 양을 상징하는 검은 돌과 흰 돌을 사용하는 게임입니다. 두 명의 플레이어는 그들의 검은 돌과 흰 돌을 각각 격자에 놓게 되고, 처음으로 5 개의 돌을 직선(혹은 대각선)으로 만든 플레이어가 승리하게 됩니다. 여러분은 이 게임의 이름을 오목이라고 부르기로 결정했습니다. 굉장히 익숙하지요? 하지만 그렇지 않습니다(...).

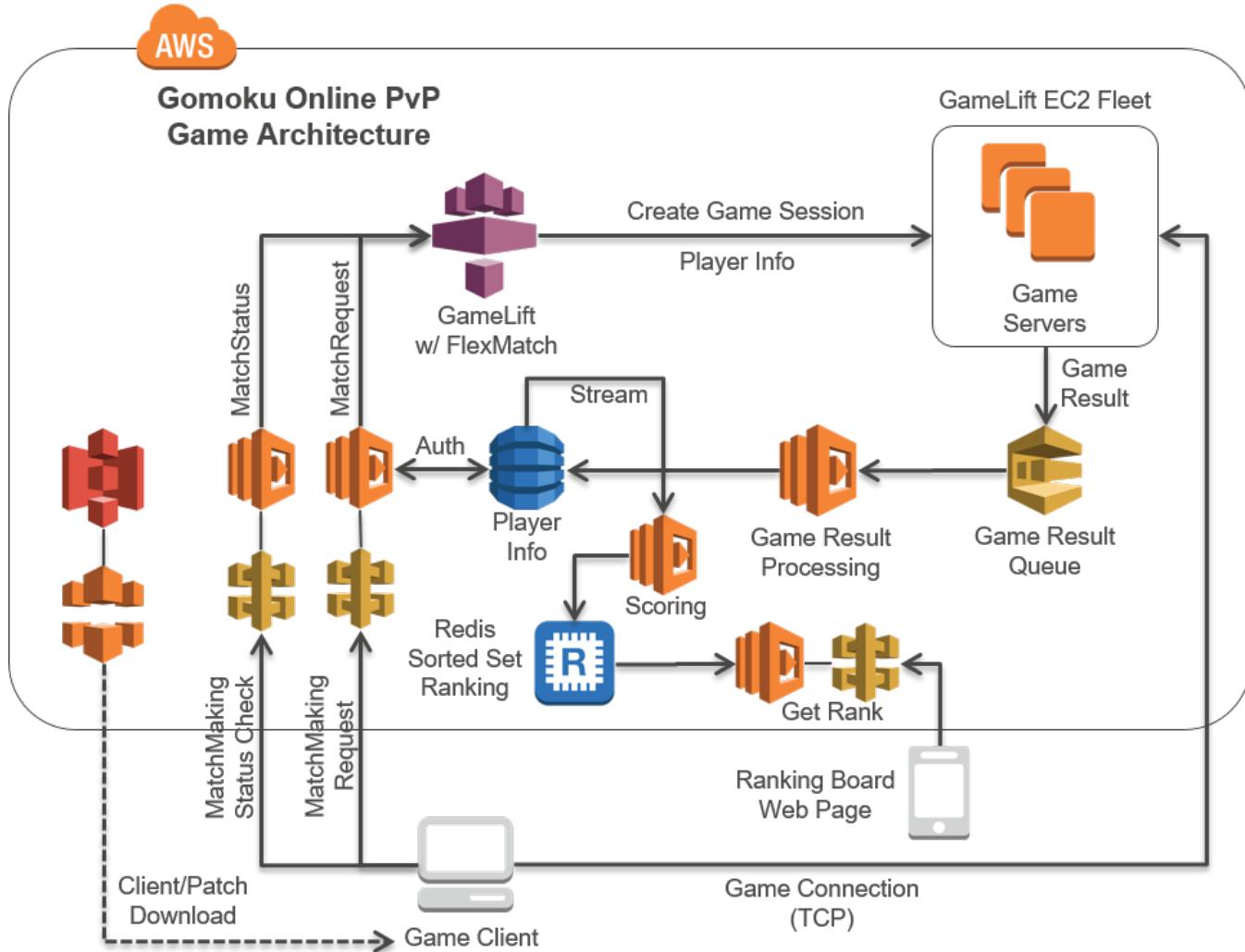
하지만 이 게임을 플레이하는 도중 사소한 오류를 발견합니다. 이 게임을 하려면 두 명의 플레이어가 있어야 하는데, 동일한 사람들과 계속 플레이를 하다 보니 게임이 재미 없다는 것입니다. 지금은 2010년대이지 1970년대가 아닙니다. 그래서 여러분은 몇 가지 AWS의 서비스를 이용하여 “인터넷”을 통한 플레이를 제공하기로 결정했습니다. 힘들 것 같나요? 하지만 게임은 이미 준비되어 있습니다.

자, 그러면 한가지만 명심하면 됩니다. 여러분에게 큰 운영(Ops) 조직이 없기 때문에 Serverless 및 관리형 서비스를 활용하여 최대한 효율적이고 최적화된 방식을 활용해서 만들어야 합니다.

즉, API Gateway, Lambda, ElastiCache(Redis), S3, DynamoDB, SQS, 그리고 GameLift Service 를 사용한 Full Stack 을 만드는겁니다!

Architecture Diagram

아래의 아키텍처는 우리가 이번 Lab 을 통해 만드는 것입니다. 본격적인 개발에 앞서 이렇게 아이디어를 아키텍처로 그리는 것이 좋습니다.



아키텍처에서 볼 수 있듯이 어느 정도의 AWS 지식만 있다면 충분히 할 수 있습니다.

Let's start to stack the full stack!

본격적인 시작에 앞서 게임 클라이언트와 서버 바이너리를 모두 다운로드 받았는지 확인하세요. 소스코드를 다운받아 직접 컴파일 할 수도 있지만, 이번 랙에서 우리는 서버와 클라이언트 모두 이미 컴파일 된 바이너리를 사용하려고 합니다.

이번 랩에서 사용될 파일의 다운로드 URL 은 따로 제공됩니다. (클라이언트 바이너리, 서버 바이너리, Lambda 코드 모두 포함되어 있습니다)

또한 랩 시작 전에 어떤 VPC 를 사용할지 결정하세요. 대부분의 서비스는 VPC 와 관계없지만 몇 가지 서비스는 VPC 에 종속성을 갖고 있습니다. 또한 우리는 보안이 뛰어난 디자인을 가져가고 싶습니다. 이번 랩에서는 default VPC 를 사용할 것이지만, 원한다면 직접 설계한 VPC 를 사용하셔도 괜찮습니다. (이번 랩은 여러분이 이미 VPC 와 보안 그룹 등을 다룰 줄 안다고 가정하고 있습니다)

그러면, 가장 중요한 Database 에서부터 시작하겠습니다.

Section 1: DynamoDB and ElastiCache – Redis and SQS. With a dash of IAM roles.

이번 랩은 DynamoDB를 사용해 사용자 정보와 대전 결과를 저장할 것입니다. 우리는 간단한 정보를 저장하고 또한 이 정보는 Web을 통해 접근할 수 있는 리더 보드에 사용될 것입니다. 추가로, DynamoDB 앞단에는 ElastiCache를 배치하여 리더 보드데이터를 캐싱하여 퍼블릭에 제공합니다. 왜냐하면 여러분의 App은 엄청난 인기를 끌게 될 것이고, 때문에 우리는 여러분의 DB가 동일한 요청을 수행하느라 부하가 걸리게 되길 원치 않습니다.

모든 게임 결과는 SQS를 이용해 대기열에 들어가고 Lamdba를 활용하여 DB에 삽입할 것입니다. 그래서 결과를 테이블에 삽입할 때까지 기다리지 않아도 됩니다. 이것은 또한 게임 흐름이 다른 구성 요소들이 끝날 때까지 기다리지 않아 플레이어에게 더 좋은 사용자 경험을 줄 수 있습니다. 수 많은 플레이어가 모두 처리해야하는 결과를 보내려고 할 때를 상상해보세요. 이것은 아마 혼란을 발생시키고 좋지 않은 사용자 경험을 주게 될 것입니다.

자, 이제 DynamoDB 테이블을 만들어 보겠습니다.

1. AWS 콘솔에 로그인하고 **DynamoDB** 페이지로 이동합니다.
<https://console.aws.amazon.com/dynamodb>
2. **리전**을 확인합니다. 랩을 진행하면서 모든 서비스 요소들을 하나의 리전에서 생성하셔야 합니다.
3. 콘솔에서 **Create Table** 을 클릭하여 DynamoDB 테이블 생성을 시작합니다.
4. Table name은 “**GomokuPlayerInfo**” 로 설정하고, Primary Key는 “**PlayerName**” 으로

Full Stack Game Demo Serverless Full Stack with Gomoku

하고 데이터 타입은 **String**을 선택합니다. 그리고 나서 **Create** 버튼을 클릭합니다.

Create DynamoDB table

DynamoDB is a schema-less database that only requires a table name and primary key. The table's primary key is made up of one or two attributes that uniquely identify items, partition the data, and sort data within each partition.

Table name* ⓘ

Primary key* Partition key

String ⓘ

Add sort key

Table settings

Default settings provide the fastest way to get started with your table. You can modify these default settings now or after your table has been created.

Use default settings

- No secondary indexes.
- Auto Scaling capacity set to 70% target utilization, at **NEW!** minimum capacity of 5 reads and 5 writes

Additional charges may apply if you exceed the AWS Free Tier levels for CloudWatch or Simple Notification Service. Advanced alarm settings are available in the CloudWatch management console.

Create

5. 테이블이 생성된 뒤에는 Stream을 활성화 합니다. 기본적으로 stream은 비활성화 되어 있습니다. Manage Stream 버튼을 클릭합니다.

GomokuPlayerInfo [Close](#)

[Overview](#) [Items](#) [Metrics](#) [Alarms](#) [Capacity](#) [Indexes](#) [Triggers](#) [Access control](#) [Tags](#) ⓘ

⌚ Table is being created

Recent alerts

No CloudWatch alarms have been triggered for this table.

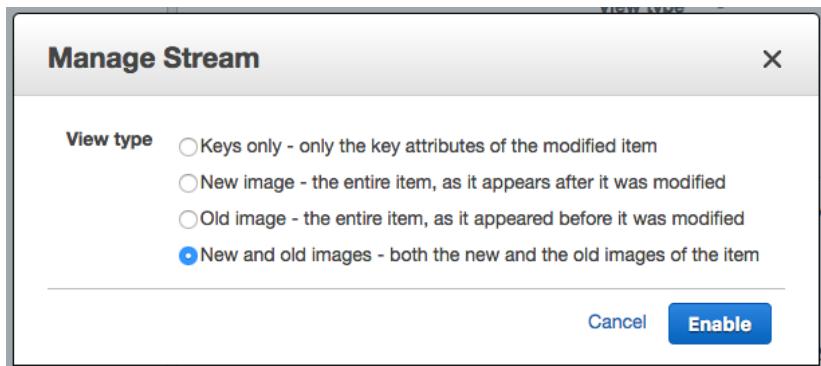
Stream details

Stream enabled	No
View type	-
Latest stream ARN	-

Manage Stream

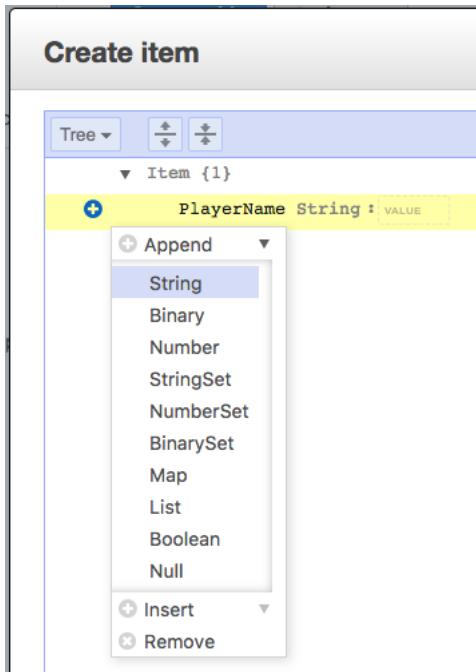
Table details

Table name	GomokuPlayerInfo
Primary partition key	PlayerName (String)
Primary sort key	-

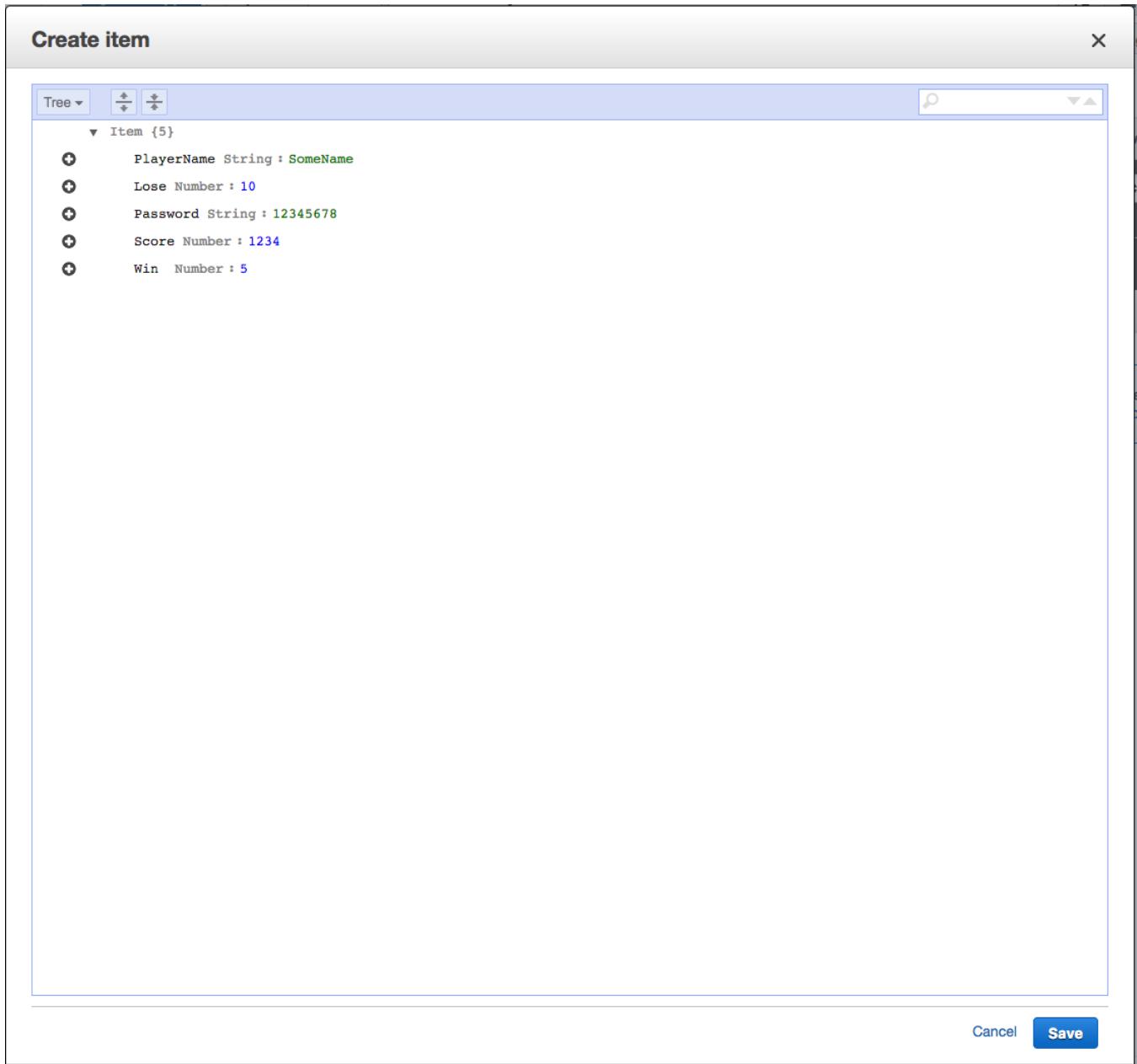


6. **New and old images** 를 선택하고 **Enable**버튼을 클릭합니다.
7. 기본적인 작업은 끝났습니다. 이제 테스트 데이터 샘플을 만들어 줍니다. 테이블에서 **Items** 탭을 선택합니다.

8. **Create Item** 을 클릭하여 새로운 항목을 만들어줍니다.
9. 편집기에서 + 버튼을 클릭하고, **Append**를 선택하여 추가합니다.



10. 다음 스크린 캡처와 같아질 때까지 데이터를 계속 추가해줍니다. 그리고 **Save** 버튼을 클릭하여 저장합니다. (데이터 타입에 주의해주세요)



(*) Items tab에서 Scan, [테이블 이름] 을 선택하고 Start search버튼을 누르면 위에서 추가한 항목이 화면이 나타나는 것을 확인 할 수 있습니다.

DynamoDB 설정은 끝났습니다.

이제 ElastiCache를 설정합니다. 이것은 순위 정보를 저장할 것입니다.

1. AWS 콘솔에서 **ElastiCache**로 이동합니다. <https://console.aws.amazon.com/elasticache>
2. ElastiCache cluster를 생성합니다. 우리는 **Redis** 엔진을 사용할 것입니다.
3. 다음 스크립 캡처와 같이 필요한 정보를 입력해줍니다.

Create your Amazon ElastiCache cluster



Cluster engine **Redis**
In-memory data structure store used as database, cache and message broker. ElastiCache for Redis offers Multi-AZ with Auto-Failover and enhanced robustness.
 Cluster Mode enabled

Memcached
High-performance, distributed memory object caching system, intended for use in speeding up dynamic web applications.

Redis settings

Name	gomokuranking	
Engine version compatibility	5.0.4	
Port	6379	
Parameter group	default.redis5.0	
Node type	cache.t2.medium (3 GiB)	
Number of replicas	0	

» Advanced Redis settings

[Cancel](#) [Create](#)

Name: gomokuranking

Engine: 5.0.x (As of 2019 June, 5.0.4)

Port: 6379 (default)

Parameter group: default

Node type: t2.medium

Number of replicas: 0

4. 모든 설정이 완료되면, **Create** 버튼을 클릭하여 Redis 클러스터를 생성합니다. (시간이 조금 걸리기 때문에 다음 단계인 SQS 생성을 먼저 진행할 수도 있습니다.)
5. 만약 특정한 VPC내에서 생성하고 싶다면 Advanced Redis setting 페이지로 이동하여 여러분의 VPC 정보를 입력합니다.
6. 생성한 Redis 클러스터의 상태가 available이 되면 Primary Endpoint를 따로 기록해둡니다. 추 후 Lambda 생성 시에 해당 Endpoint가 사용됩니다.

The screenshot shows the AWS ElastiCache console interface. At the top, there are buttons for Create, Backup, Reboot, Delete, and Modify. Below that, a table displays the 'gomokuranking' cluster information. The table columns include Cluster Name, Mode, Shards, Nodes, Node Type, and Status. The cluster details are as follows:

Cluster Name	Mode	Shards	Nodes	Node Type	Status
gomokuranking	Redis	0	1 node	cache.t2.medium	available

Configuration details for the cluster:

- Name: gomokuranking
- Creation Time: August 28, 2017 at 2:38:51 PM UTC+9
- Configuration Endpoint: -
- Status: available
- Primary Endpoint: gomokuranking.ybsuea.0001.usw2.cache.amazonaws.com:6379
- Engine: Redis
- Engine Version Compatibility: 3.2.4
- Node type: cache.t2.medium
- Availability Zones: us-west-2a
- Shards: 0
- Number of Nodes: 1 node
- Multi-AZ: Disabled
- Description: -
- Parameter Group: default.redis3.2 (in-sync)
- Subnet Group: default
- Maintenance Window: wed:13:00-wed:14:00
- Notification ARN: Disabled
- Backup Retention Period: Disabled
- Backup Window: Disabled

7. 생성한 ElastiCache의 보안을 강화하기 위하여 gomokuranking 클러스터에 안전한 보안 그룹을 생성하여 할당합니다. 이번 실습에서는 아주 간단한 보안그룹 정책을 생성하여 할당하겠습니다.
8. VPC 콘솔에서 <https://console.aws.amazon.com/vpc> 좌측의 **Security Group** 메뉴를 선택합니다..
9. **Create Security Group** 버튼을 클릭합니다.

The dialog box is titled "Create Security Group". It contains the following fields:

- Name tag: GomokuDefault
- Group name: GomokuDefault
- Description: Domoku Default
- VPC: vpc-6fdade08 | defaultVPC

At the bottom right, there are "Cancel" and "Yes, Create" buttons. The "Yes, Create" button is highlighted in blue.

Full Stack Game Demo
Serverless Full Stack with Gomoku

10. Name tag, Group name 등에 적절한 정보를 입력해주고, VPC는 실습을 진행 중인 **default VPC**를 선택합니다. (리전에 VPC가 하나라면 default가 따로 표시되지는 않습니다)
11. 보안 그룹 내의 통신을 위하여 inbound 정책을 수정해야 합니다. 생성한 보안 그룹을 선택하고 **Inbound Rules**탭을 클릭합니다.

The screenshot shows the AWS Security Groups console. At the top, there are tabs for 'Create Security Group' and 'Security Group Actions'. A search bar contains the text 'gomoku'. Below the search bar is a filter section with dropdowns for 'Name tag', 'Group ID', 'Group Name', 'VPC', and 'Description'. A single entry is listed: 'GomokuDefault' (sg-d1406bab). The main area shows the security group details for 'sg-d1406bab | GomokuDefault'. Below this, there are tabs for 'Summary', 'Inbound Rules' (which is selected), 'Outbound Rules', and 'Tags'. Under the 'Edit' tab, there are four sub-tabs: 'Type', 'Protocol', 'Port Range', and 'Source'. A message says 'You do not have any Inbound Rules.'.

12. **Edit** 버튼을 클릭하고 다음 스크린 캡처와 같이 정책을 생성합니다. 여기서 **Source**에는 보안 정책 **자신의 Group ID**를 입력합니다. 이렇게 함으로써 이 보안 그룹을 할당한 호스트와 서비스들끼리 통신을 할 수 있습니다. 생성한 보안 그룹을 기억해 둡니다.

The screenshot shows the AWS Security Groups console with the 'Edit' tab selected. The 'Inbound Rules' tab is also selected. At the top, there are tabs for 'Cancel' and 'Save'. Below these are four columns: 'Type', 'Protocol', 'Port Range', and 'Source'. The 'Type' dropdown is set to 'ALL Traffic', 'Protocol' to 'ALL', 'Port Range' to 'ALL', and 'Source' to 'sg-d1406bab'. There is a 'Remove' button next to the source field. At the bottom, there is a button labeled 'Add another rule'.

Type: All traffic

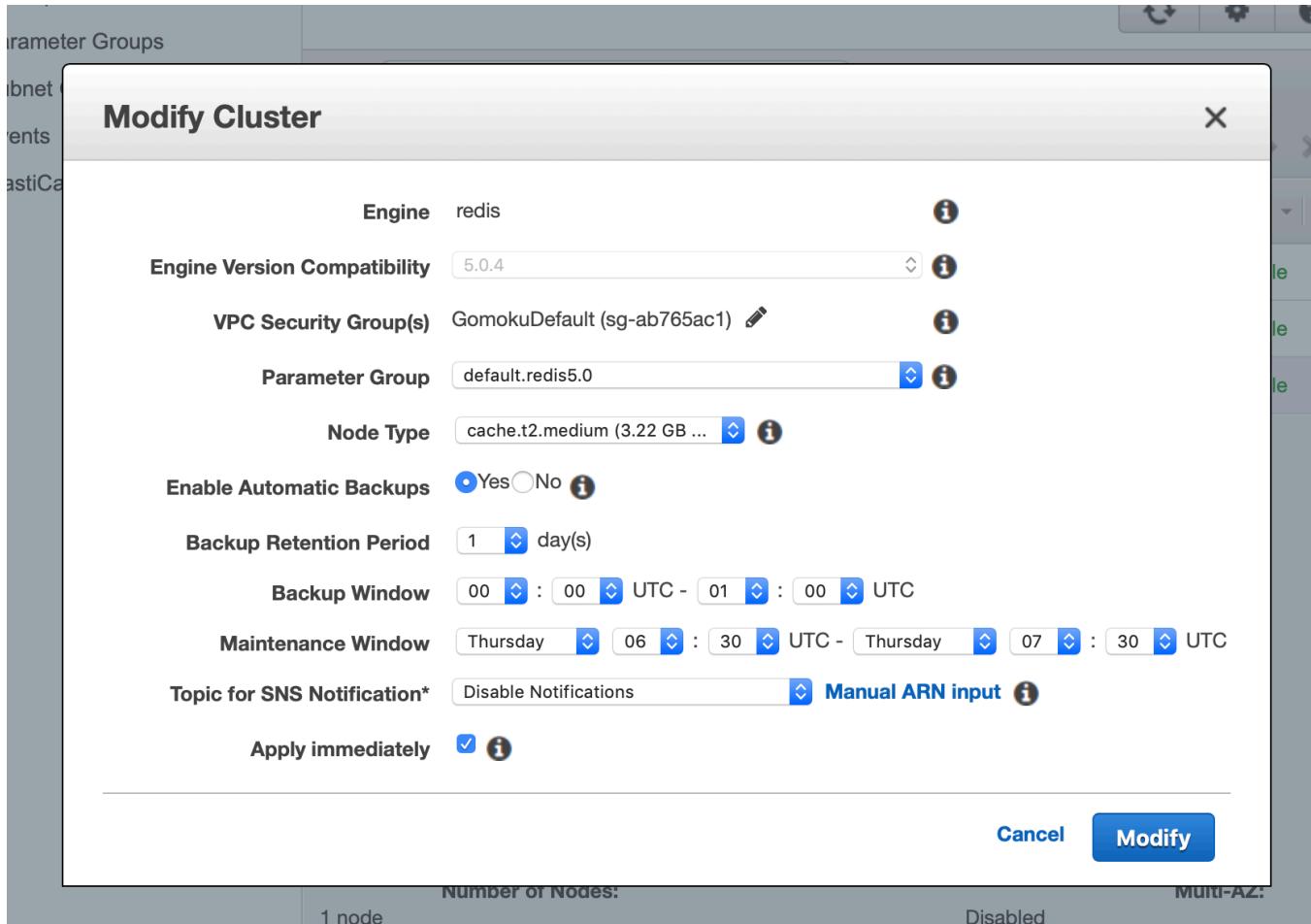
Protocol: All

Source: Security Group itself

13. 보안 그룹 생성을 완료하였다면 다시 ElastiCache 페이지로 돌아와 생성한 Redis

클러스터를 선택합니다.

14. 클러스터를 선택하고 상단의 **Modify** 버튼을 클릭합니다.



15. 팝업 메뉴에서 VPC Security Group에 방금 생성한 보안그룹을 선택한 뒤 Modify 버튼을 클릭하여 완료합니다.

ElastiCache 설정을 완료하였습니다. 이제 SQS 설정을 시작합니다. SQS를 이용하여 게임 결과 처리를 위한 대기열을 만들 것입니다.

1. 콘솔에서 SQS 메뉴로 들어갑니다. <https://console.aws.amazon.com/sqs>
2. **Create New Queue**를 클릭하여 생성을 시작합니다. Queue 이름은 **game-result-queue**로 입력하고, Standard Queue를 선택합니다. **Quick-Create Queue** 버튼을

Full Stack Game Demo
Serverless Full Stack with Gomoku

클릭하여 Queue 생성을 완료합니다.

Create New Queue

What do you want to name your queue?

Queue Name

game-result-queue

Region Asia Pacific (Seoul)

What type of queue do you need?

Standard Queue

FIFO Queue

Unlimited Throughput: Standard queues support a nearly unlimited number of transactions per second (TPS) per API action.

At-Least-Once Delivery: A message is delivered at least once, but occasionally more than one copy of a message is delivered.

Best-Effort Ordering: Occasionally, messages might be delivered in an order different from which they were sent.

High Throughput: FIFO queues support up to 300 messages (300 send, receive, or delete operations per second). When you messages per operation (maximum), FIFO queues can support messages per second. To request a limit increase, file a suppo

First-In-First-out Delivery: The order in which messages are received is strictly preserved.

Exactly-Once Processing: A message is delivered once and available until a consumer processes and deletes it. Duplicates introduced into the queue.



- Queue 생성이 완료되면 Details에 보이는 endpoint URL을 기록해둡니다. 마찬가지로 뒤의 Lambda 소스 코드에 사용할 예정입니다.

Filter by Prefix: Enter Text... X

1 to 1 of 1 items > < 1

Name	Queue Type	Content-Based Deduplication	Messages Available	Messages in Flight	Created
game-result-queue	Standard	N/A	0	0	2017-08-28 14:40:33 GMT+09:00

1 SQS Queue selected

Details Permissions Redrive Policy Monitoring Encryption

Details

- Name: game-result-queue
- URL: <https://sqs.us-west-2.amazonaws.com/123456789012/game-result-queue>
- ARN: arn:aws:sqs:us-west-2:123456789012:game-result-queue
- Created: 2017-08-28 14:40:33 GMT+09:00
- Last Updated: 2017-08-28 14:40:33 GMT+09:00
- Delivery Delay: 0 seconds
- Queue Type: Standard
- Content-Based Deduplication: N/A

Default Visibility Timeout: 30 seconds
Message Retention Period: 4 days
Maximum Message Size: 256 KB
Receive Message Wait Time: 0 seconds
Messages Available (Visible): 0
Messages in Flight (Not Visible): 0
Messages Delayed: 0

생성된 Queue와 함께 아까 생성된 Dynamo DB, ElastiCache의 동작은 뒤의 Lambda를 통하여 확인하도록 하겠습니다.

마지막으로, 이번 실습에서 만드는 full stack application에 사용할 IAM 정책(policy)와 역할(role)을 생성합니다.

1. 우선, 역할에 사용될 정책을 만듭니다. 이 랙에서는 편의를 위해서 AWS Managed policy를 주로 사용합니다만, GameLift 서비스는 현재 managed policy를 제공하지 않으므로 명시적으로 정책을 생성해야 합니다. 콘솔에서 **IAM** 메뉴로 이동합니다.
<https://console.aws.amazon.com/iam>
2. 메뉴에서 **Policies**를 선택하고 **Create policy** 버튼을 클릭합니다.
3. Visual editor에서 **Service**는 GameLift를 선택하고 **Actions**에서는 랩의 편의상 All GameLift actions를 선택합니다. **Review Policy**를 클릭하여 다음 단계로 진행합니다.

Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor JSON Import managed policy

Expand all | Collapse all

▼ GameLift (All actions) Clone | Remove

► Service GameLift

▼ Actions Specify the actions allowed in GameLift [?](#) Switch to deny permissions [!](#)

close Filter actions

Manual actions [\(add actions\)](#)

All GameLift actions (gamelift:*)

Access level

► List (3 selected)

► Read (19 selected)

► Write (18 selected)

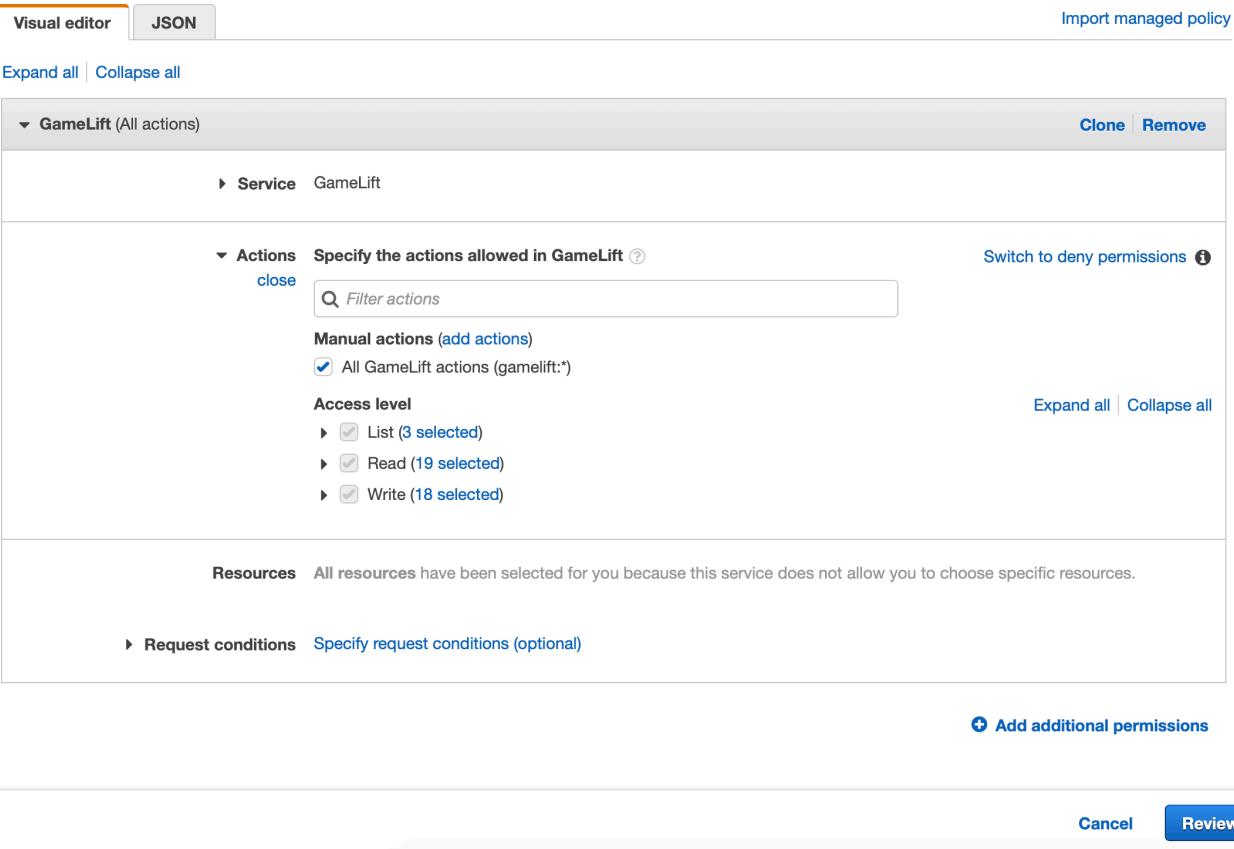
Expand all | Collapse all

Resources All resources have been selected for you because this service does not allow you to choose specific resources.

► Request conditions Specify request conditions (optional)

[+ Add additional permissions](#)

Cancel Review policy



4. 정책의 이름을 입력하고 **Create policy** 버튼을 클릭합니다.

Create policy

Review policy

Name* GameLiftFullAccess|
Use alphanumeric and '+=, @-_ characters. Maximum 128 characters.

Description
Maximum 1000 characters. Use alphanumeric and '+=, @-_ characters.

Summary

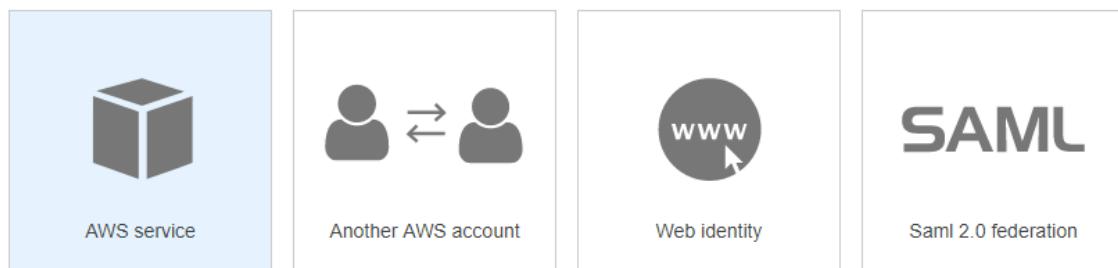
Service	Access level	Resource	Request condition
Allow (1 of 189 services) Show remaining 188	GameLift	Full access	All resources
			None

* Required [Cancel](#) [Previous](#) **Create policy**

5. 다음은 후반부 Lamdba 함수에 사용할 5 가지 역할(Role)을 생성할 것입니다.
6. IAM 콘솔에서 **Role** 메뉴로 이동한 뒤 **Create role** 버튼을 클릭합니다.
7. Role type에서 AWS Service Role의 **AWS Lambda**를 선택합니다.

Full Stack Game Demo
Serverless Full Stack with Gomoku

Select type of trusted entity



Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

API Gateway	Data Pipeline	IoT	Service Catalog
Auto Scaling	Directory Service	Lambda	
Batch	DynamoDB	Lex	
CloudFormation	EC2	Machine Learning	
CloudHSM	EC2 Container Service	OpsWorks	
CloudWatch Events	EMR	RDS	
CodeBuild	Elastic Beanstalk	Redshift	
CodeDeploy	Elastic Transcoder	SMS	
Config	Glue	SNS	
DMS	Greengrass	SWF	

Select your use case

Lambda

Allows Lambda functions to call AWS services on your behalf.

8. 이 Role에는 3개의 정책을 할당할 것입니다. 첫 번째는 **AmazonSQSFullAccess** 정책입니다.

Create role

1 2 3 4

▼ Attach permissions policies

Choose one or more policies to attach to your new role.

Create policy			
Filter policies ▾		<input type="text"/> Q: sqs	
Showing 3 results			
	Policy name ▾	Used as	Description
<input checked="" type="checkbox"/>	▶ AmazonSQSFullAccess	Permissions policy (1)	Provides full access to Amazon SQS via ...
<input type="checkbox"/>	▶ AmazonSQSReadOnlyAccess	None	Provides read only access to Amazon S...
<input type="checkbox"/>	▶ AWSLambdaSQSQueueExecutionRole	None	Provides receive message, delete messa...

9. 그 다음에는 **AmazonDynamoDBFullAccess** 정책 및 **AWSLambdaBasicExecutionRole** 정책을 선택한 뒤 **Next: Tag** 버튼을 클릭합니다.

Create role

1 2 3 4

▼ Attach permissions policies

Choose one or more policies to attach to your new role.

[Create policy](#)



Filter policies ▾		Policy name ▾	Used as	Description
<input checked="" type="checkbox"/>	▶	AmazonDynamoDBFullAccess	None	Provides full access to Amazon Dyna...
<input type="checkbox"/>	▶	AmazonDynamoDBFullAccesswithDataPipeline	None	Provides full access to Amazon Dyna...
<input type="checkbox"/>	▶	AmazonDynamoDBReadOnlyAccess	None	Provides read only access to Amazon ...
<input type="checkbox"/>	▶	aws-iot-role-dynamoPut_-1682581677	Permissions policy (1)	
<input type="checkbox"/>	▶	aws-iot-role-dynamoPut_1796729520	Permissions policy (1)	
<input type="checkbox"/>	▶	AWSApplicationAutoscalingDynamoDBTableP...	Permissions policy (1)	Policy granting permissions to applica...

10. Tag는 선택 사항으로 입력합니다. **Next: Review**를 선택합니다. Role name은 **Gomok-game-sqs-process**를 입력합니다. (AWS Console의 최신 UI 업데이트 상황에 따라 일부 웹 인터페이스 변경되었을 수 있습니다.)

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name*

Gomok-game-sqs-process

Use alphanumeric and '+=_,@-' characters. Maximum 64 characters.

Role description

Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=_,@-' characters.

Trusted entities AWS service: lambda.amazonaws.com

Policies

AmazonSQSFullAccess

AmazonDynamoDBFullAccess

AWSLambdaBasicExecutionRole

Permissions boundary Permissions boundary is not set

No tags were added.

11. **Create role** 버튼을 클릭하여 첫 번째 역할 생성을 완료합니다.
12. 두 번째 역할도 첫 번째와 동일한 방법으로 생성합니다. 하지만 이번에는 **AmazonDynamoDBFullAccess**, **AmazonVPCFullAccess**, **AWSLambdaBasicExecutionRole** 정책을 추가합니다.
13. 두 번째 역할의 이름은 **Gomok-game-rank-update**로 지정합니다.
14. 세 번째 Role도 동일한 방법으로 생성합니다. 이번에는 **VPC Full Access**, **AWSLambdaBasicExecutionRole** 정책을 추가해 줍니다.
15. 세 번째 역할의 이름은 **Gomok-game-rank-reader**로 지정합니다.
16. 네 번째 역할도 첫번째와 동일한 방법으로 생성합니다. 이번에는 **AWSLambdaBasicExecutionRole**, **AmazonDynamoDBFullAccess**, 그리고 앞에서 생성한 **GameLiftFullAccess** 정책을 추가해 줍니다.
17. 네 번째 역할의 이름은 **Gomok-game-match-request**로 지정합니다.
18. 마지막으로 다섯 번째 역할도 동일한 방법으로 생성합니다. 이번에는 **AWSLambdaBasicExecutionRole**과 앞에서 생성한 **GameLiftFullAccess** 정책을 추가해 줍니다.
19. 다섯 번째 역할의 이름은 **Gomok-game-match-status**로 지정합니다.
20. 모두 생성이 완료되었다면 다음 스크린 캡처와 같이 다섯개의 역할을 확인할 수 있습니다.

The screenshot shows the AWS Lambda Roles list. At the top, there are buttons for 'Create role' and 'Delete role'. To the right are three small icons: a refresh symbol, a gear symbol, and a question mark symbol. Below these are two search/filter fields: one for 'Role name' containing 'gomok' and another for 'Status' which is set to 'Showing 5 results'. The main list area contains five entries, each with a checkbox and a link:

- Gomok-game-match-request
- Gomok-game-match-status
- Gomok-game-rank-reader
- Gomok-game-rank-update
- Gomok-game-sqs-process

21. 마지막으로 하나의 역할을 더 생성합니다. 앞서는 Lambda가 사용할 역할을 생성하였지만, 이번에는 GameLift에서 생성하는 Fleet에서 사용할 역할을 생성합니다. GameLift의 Fleet에서는 Game의 결과를 SQS에 전송할 때 사용하는 권한이 필요합니다.
22. Create Role을 선택하여 새로운 역할을 생성합니다. 이 역할의 사용은 GameLift

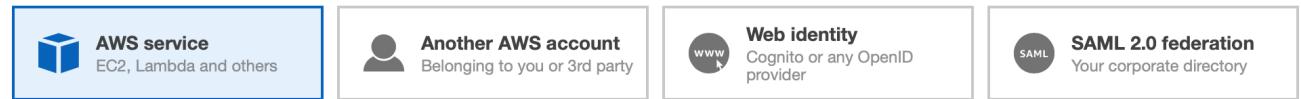
Full Stack Game Demo
Serverless Full Stack with Gomoku

서비스가 수행하므로 GameLift를 서비스를 선택해야 하지만, 콘솔에는 GameLift가 서비스로 표시되지 않습니다. 여기서는 EC2를 선택하고 다음으로 이동합니다.

Create role

1 2 3 4

Select type of trusted entity



Allows AWS services to perform actions on your behalf. [Learn more](#)

Choose the service that will use this role

EC2

Allows EC2 instances to call AWS services on your behalf.

Lambda

Allows Lambda functions to call AWS services on your behalf.

API Gateway	Comprehend	ElastiCache	Lex	SMS
AWS Backup	Config	Elastic Beanstalk	License Manager	SNS
AWS Support	Connect	Elastic Container Service	Machine Learning	SWF
Amplify	DMS	Elastic Transcoder	Macie	SageMaker
AppSync	Data Lifecycle Manager	Elastic Load Balancing	MediaConvert	Security Hub
Application Auto Scaling	Data Pipeline	Forecast	Migration Hub	Service Catalog
Application Discovery Service	DataSync	Glue	OpsWorks	Step Functions
Batch	DeepLens	Greengrass	Personalize	Storage Gateway
CloudFormation	Directory Service	GuardDuty	RAM	Textract
CloudHSM	DynamoDB	Inspector	RDS	Transfer
CloudTrail	EC2	IoT	Redshift	Trusted Advisor
CloudWatch Application Insights	EC2 - Fleet	IoT Things Graph	Rekognition	VPC
CloudWatch Events	EC2 Auto Scaling	KMS	RoboMaker	WorkLink
CodeBuild	EKS	Kinesis	S3	WorkMail
CodeDeploy	EMR	Lambda		

* Required

Cancel

Next: Permissions

23. 이 역할이 필요한 권한은 AmazonSQSFullAccess를 선택합니다.

Create role

1 2 3 4

▼ Attach permissions policies

Choose one or more policies to attach to your new role.

[Create policy](#)



Filter policies ▾		Policy name ▾	Used as	Description
<input checked="" type="checkbox"/>	▶	AmazonSQSFullAccess	Permissions policy (3)	Provides full access to Amazon SQS v...
<input type="checkbox"/>	▶	AmazonSQSReadOnlyAccess	None	Provides read only access to Amazon ...
<input type="checkbox"/>	▶	AWSLambdaSQSQueueExecutionRole	None	Provides receive message, delete mes...
<input type="checkbox"/>	▶	AWSQuicksightAthenaAccess	Permissions policy (1)	Quicksight access to Athena API and ...
<input type="checkbox"/>	▶	AWSQuickSightDescribeRDS	None	Allow QuickSight to describe the RDS ...
<input type="checkbox"/>	▶	AWSQuickSightDescribeRedshift	None	Allow QuickSight to describe Redshift ...
<input type="checkbox"/>	▶	AWSQuickSightIAMPolicy	Permissions policy (1)	Grants Amazon QuickSight list permis...
<input type="checkbox"/>	▶	AWSQuickSightIoTAnalyticsAccess	None	Give QuickSight read-only access to I...

▶ Set permissions boundary

* Required

[Cancel](#)

[Previous](#)

[Next: Tags](#)

24. 역할 이름은 Gomoku-GameLiftFleetRole로 입력합니다.

Full Stack Game Demo Serverless Full Stack with Gomoku

Create role

1 2 3 4

Review

Provide the required information below and review this role before you create it.

Role name* Gomoku-GameLiftFleetRole

Use alphanumeric and '+,-,@-' characters. Maximum 64 characters.

Role description

Allows EC2 instances to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+,-,@-' characters.

Trusted entities AWS service: ec2.amazonaws.com

Policies AmazonSQSFullAccess

Permissions boundary Permissions boundary is not set

No tags were added.

* Required

Cancel

Previous

Create role

25. 이 역할을 사용할 수 있는 서비스를 GameLift로 변경해야 합니다. 앞서 생성한 역할을 선택하여 Trust relationships 탭을 선택하고, Edit trust relationship 버튼을 클릭합니다.
26. 아래와 같이 ec2로 되어 있던 부분을 gamelift로 변경하고 update trust policy 버튼을 클릭하여 저장합니다.

Edit Trust Relationship

You can customize trust relationships by editing the following policy document.

Policy Document

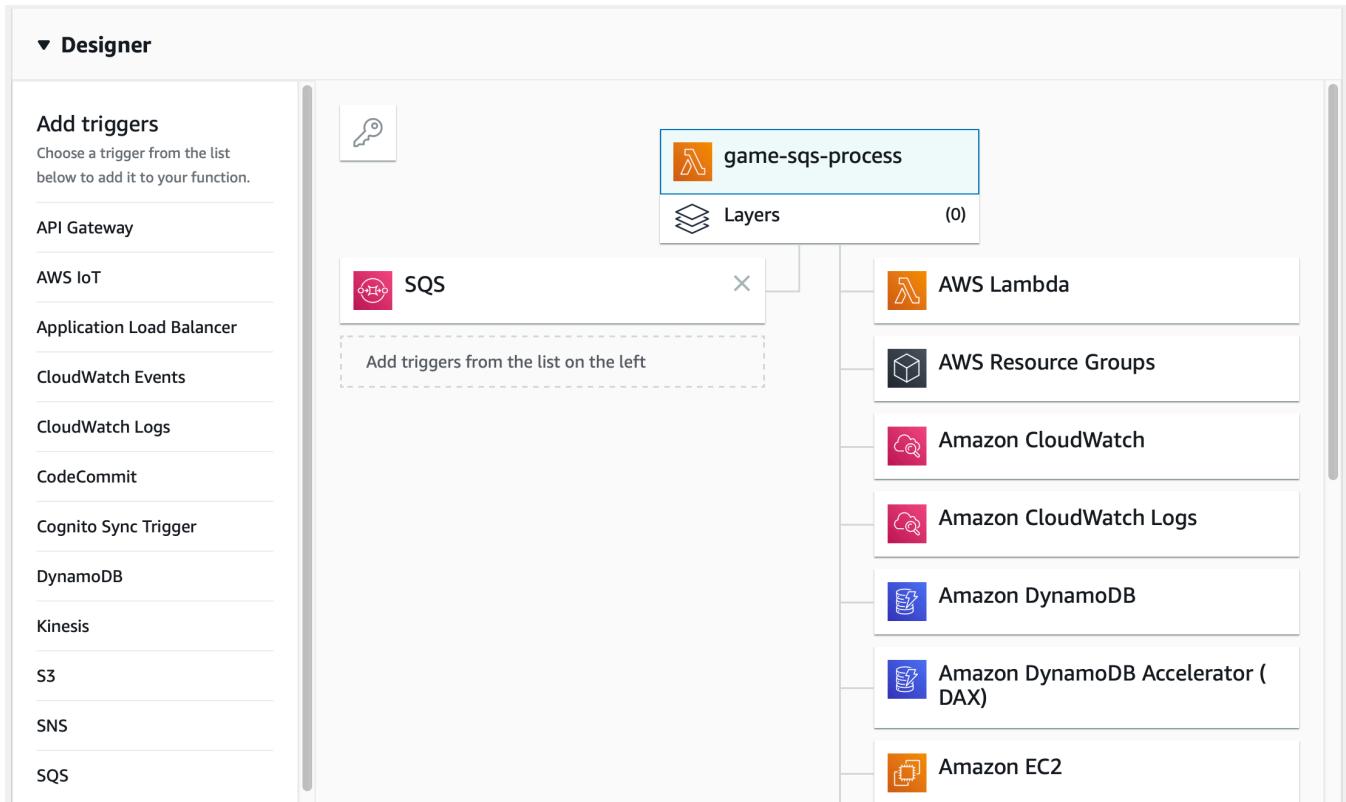
```
1 {  
2   "Version": "2012-10-17",  
3   "Statement": [  
4     {  
5       "Effect": "Allow",  
6       "Principal": {  
7         "Service": "gamelift.amazonaws.com"  
8       },  
9       "Action": "sts:AssumeRole"  
10      }  
11    ]  
12  }
```

Section 2: Lambda Lambda Lambda

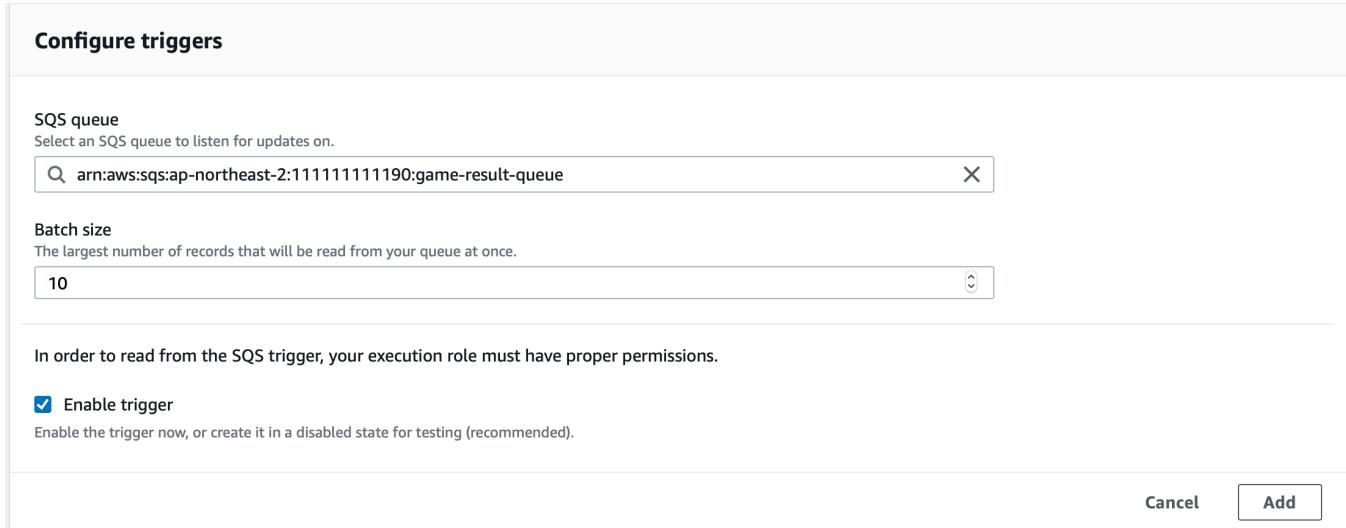
이전 섹션까지 우리는 full game stack의 가장 기초적인 서비스들을 구성했습니다. 지금부터는 사용자가 게임을 즐기고 게임 결과를 처리할 Lambda 를 구성하겠습니다.

이번 랙에서는 총 3개의 Lambda 함수를 생성할 것입니다.

1. 콘솔에서 **Lambda** 메뉴로 이동합니다. <https://console.aws.amazon.com/lambda>
2. **Create function** 버튼을 클릭하여 첫번째 함수 생성을 시작합니다.
3. **Author from scratch** 메뉴를 선택하여 빈 함수를 우선 생성합니다.
4. **Name** 항목에는 **game-sqs-process**를 입력합니다.
5. **Runtime**은 Python 2.7을 선택합니다.
6. **Permissions** 항목에서 **Role** 은 Use an existing role을 선택하고 기존에 만들어둔 **Gomok-game-sqs-process**를 선택하고 Create function을 실행합니다.
7. 생성이 완료되면 Designer그룹의 Add triggers 하단에 SQS(Simple Queue Service)를 Lambda의 실행 트리거로 선택합니다. 그러면 하단에 Configure triggers 그룹이 생성됩니다.



8. SQS queue는 앞서 생성한 SQS의 arn이 선택되어 있는 것을 확인합니다. Enable trigger가 선택된 것을 확인하고 Add 버튼을 선택합니다.



9. 다시 상단의 Designer 그룹으로 돌아와 아래의 생성할 함수의 정보를 참고하여 Lambda 함수를 작성합니다. (GameLiftFlexHol.zip 파일을 압축해제하면 Lambda 폴더 밑에 있는 GameResultProcessing.py에서 표시된 변수를 적합하게 수정해서 넣어주는 과정입니다)

Code: GameResultProcessing.py 파일의 내용을 Copy&Paste합니다. 코드 내의 **region_name** 부분은 여러분이 랩을 수행하는 리전으로 되어 있는지 확인합니다. (예: ap-northeast-2)

Advanced settings: 128MB Memory and **0 min 30 sec timeout**

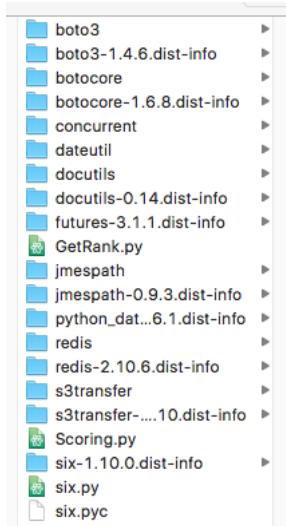
region_name : 여러분이 사용하시는 리전의 코드입니다.

10. Save 버튼을 클릭하여 함수를 생성합니다. 생성한 함수는 SQS에 기록된 게임 결과 점수를 읽어와 DynamoDB에 업데이트하는 역할을 수행합니다.

두 번째 Lambda 부터는 앞의 Lambda와는 다르게 Python 배포 패키지를 통째로 업로드 하여 Lambda 함수를 생성할 것입니다. 이전에 생성한 함수는 Python 표준 SDK만을 사용하기 때문에 인라인 편집기를 사용했지만, 다른 Lambda 함수들은 Redis 라이브러리를 참조하기 때문에 배포 패키지를 업로드 하여 함수를 생성할 것입니다. Lambda 함수를 Python 배포 패키지를 업로드하여 생성해볼 수 있는 아주 좋은 기회입니다.

소스 코드를 작성(수정)한 후 배포 패키지 형태로 묶은 다음(LambdaDeploy.zip 파일) 이를 업로드하여 Lambda 함수를 생성할 것입니다. (만약 배포 패키지를 어떻게 직접 만드는지 알고 싶으시다면 Appendix A를 참조하세요.)

1. GameLiftFlexHol.zip 파일의 Lambda 폴더 하위에 LambdaDeploy 폴더에 GetRank.py와 Scoring.py라는 두 개의 Python 파일이 보일 것입니다.



2. 우선 **Scoring.py** 파일을 열고 6번째 줄의 Redis 클러스터 정보를 여러분이 생성한 ElastiCache의 Endpoint로 수정합니다.
3. 두 번째로 **GetRank.py** 파일을 열고 6번째 줄의 Redis 클러스터 정보를 생성한 ElastiCache의 Endpoint로 수정합니다.
4. 두 개의 파일 모두 저장한 뒤, 다시 LambdaDeploy.zip으로 압축해줍니다. (참고: "LambdaDeploy" 폴더가 압축파일에 포함되면 안됩니다. 즉, GetRank.py 및 Scoring.py 파일은 압축파일 내의 루트경로에 있어야 합니다.)

배포 패키지를 완성하였습니다. 이제 이를 이용하여 Lambda 함수를 생성하겠습니다.

1. 앞 서와 동일하게 **Author from scratch** 메뉴를 선택하여 함수 생성을 시작합니다.
2. Name은 **game-rank-update**으로 지정하고, Runtime은 **Python 2.7**로 선택하고, Role은 **Gomok-game-rank-update**을 선택하고 Create function을 누릅니다.
3. Function code 그룹에서 Code entry type을 **Upload a .ZIP file**로 선택하고 **LambdaDeploy.zip**을 업로드합니다.

4. Handler 항목에서는 **Scoring.handler** 를 입력합니다.
5. 하단의 Basic settings 그룹에서 Timeout을 1분으로 변경합니다.

Basic settings

Description

Memory (MB) [Info](#)
Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout [Info](#)

1

:

0

sec

6. Network 항목에서 VPC를 기준에 생성한 ElastiCache가 있는 VPC로 선택합니다.
subnet항목은 모두 선택하여 주시고, Security Group은 이전에 생성했던
GomokuDefault로 선택하여 주시기 바랍니다.

Network

VPC Info
Select a VPC that your function will access.

Default `Default` (172.31.0.0/16) | default ▾

Subnets*
Select the VPC Subnets that Lambda should use to set up your VPC configuration. Format: "subnet-id (cidr-block) | az name-tag".

▼

subnet-`fa50b0b6` (172.31.16.0/20) | ap-northeast-2c def-pub-2c X

subnet-`bd3a8bd5` (172.31.0.0/20) | ap-northeast-2a def-pub-2a X

Security Groups*
Select the VPC Security Groups that Lambda should use to set up your VPC configuration. Format: "sg-id (sg-name) | name-tag".
The table below will show the inbound and outbound rules for the security groups you have selected.

▼

sg-`ab765ac1` (GomokuDefault) | GomokuDefault X

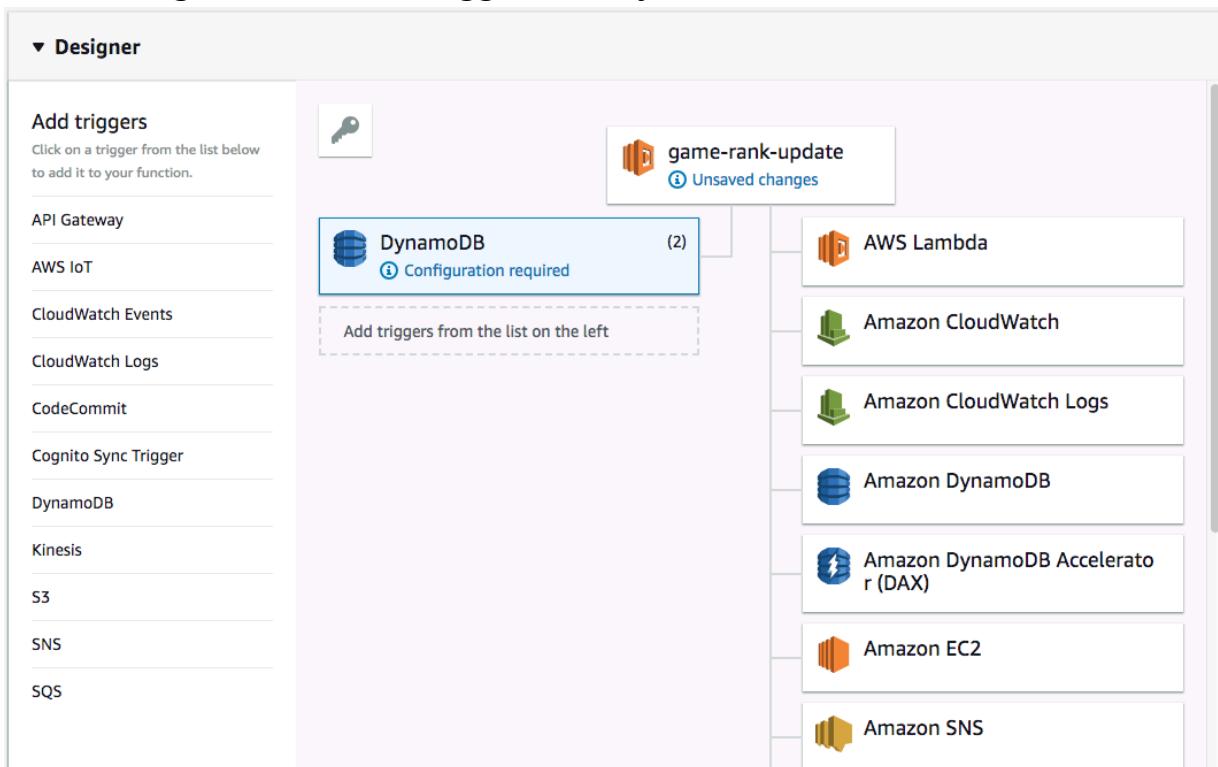
When you enable VPC, your Lambda function will lose default internet access. If you require external internet access for your function, ensure that your security group allows outbound connections and that your VPC has a NAT gateway.

Inbound rules | Outbound rules

< 1 >

Security group ID	Ports	Source
sg- <code>ab765ac1</code>	All	sg- <code>ab765ac1</code>

7. 상단의 Designer 그룹의 Add triggers에서 **DynamoDB**을 선택합니다.



8. 하단에 Configure triggers 그룹에서 **GomokuPlayerInfo** DynamoDB 테이블을 Trigger 테이블로 사용할 것입니다. 다른 부분은 기본값을 유지하고 Starting position은 **Trim horizon**을 선택합니다. **Enable trigger**를 체크한 뒤 **Add** 버튼을 클릭합니다.

Configure triggers

DynamoDB table
Select a DynamoDB table to listen for updates on.

Batch size
The largest number of records that will be read from your table's update stream at once.

Starting position
The position in the stream to start reading from. For more information, see [ShardIteratorType](#) in the Amazon DynamoDB Streams API Reference.

In order to read from the DynamoDB trigger, your execution role must have proper permissions.

Enable trigger
Enable the trigger now, or create it in a disabled state for testing (recommended).

- 위의 과정이 완료되면 오른쪽 상단의 **Save**를 눌러 함수 생성을 완료합니다. (이 단계에서 함수 Test를 누르게 되면 아직 실패합니다.)

마지막으로 세 번째 Lambda 함수를 생성하겠습니다.

- 세 번째 함수도 **Author from scratch** 를 선택하여 생성합니다.
- Name 항목에서 **game-rank-reader**를 입력하고, Runtime을 Python 2.7로 설정하고 Role을 이전에 만들어둔 **Gomok-game-rank-reader**를 선택합니다.
- Create function을 누른 후 Code entry type에서 Upload a .ZIP file을 선택하여 LambdaDeploy.zip을 업로드합니다.
- Handler 부분에서 **GetRank.handler**를 입력합니다.
- Basic settings 부분에서 Timeout을 1 min으로 설정합니다.
Network 부분에서 이전 함수 동일하게 VPC 를 기준에 생성한 ElastiCache 가 있는 VPC 로 선택합니다. subnet 항목은 모두 선택하여 주시고, Security Group 은 이전에 생성했던 GomokuDefault 로 선택하여 주시기 바랍니다.
- 이제 Save 를 눌러 Lambda 함수를 생성합니다.

여기까지 완료했다면 다음 스크린캡처와 같이 3개의 Lambda 함수가 생성된 것을 확인할 수 있습니다.

Functions (8)					View details	Test	Delete	Create function
<input type="text"/> Add filter								
keyword : game ×								
	Function name	Description	Runtime	Code size	Last Modified			
○	game-rank-update		Python 2.7	609 bytes	1 minute ago			
○	game-rank-reader		Python 2.7	441 bytes	2 minutes ago			
○	game-sqs-process		Python 2.7	923 bytes	15 hours ago			

이제 모든 Lambda 함수는 준비되었습니다. 지금부터는 이 함수를 실행할 방법이 필요합니다. 처음 생성한 두 함수는 SQS와 DynamoDB에 의해 호출될 것입니다. 하지만 마지막 함수는 trigger를 설정하지 않았습니다. 어떻게 해야 해당 함수를 호출할 수 있을까요? 여기에서 API Gateway를 사용할 것입니다. 이 부분은 다음 섹션에서 내용이 이어집니다.

여기까지 준비가 되면 Section 1과 2에서 만들어진 여러 요소들이 어떻게 동작하는지를 테스트해 볼 수 있습니다. 간략하게 설명하면 전체 흐름을 보면 SQS에 데이터가 삽입되면 만들어진 Lambda함수가 실행되면서 해당 SQS데이터를 DynamoDB에 업데이트합니다. Dynamo DB에 입력된 데이터는 DynamoDB Stream의 Trigger를 통하여 다른 Lambda함수가 실행되어 Redis Cache가 업데이트 되는 형태를 가지게 됩니다.

간단하게 Redis Cache의 내용을 확인할수는 없으나 전체 흐름이 정상적으로 동작하는지 여부를 테스트 데이터의 입력과 CloudWatch를 통하여 확인해보도록 하겠습니다.

1. Console을 통하여 SQS를 열어줍니다. <https://console.aws.amazon.com/sqs>
2. 위에서 생성한 **game-result-queue**를 선택하고 화면 위의 **Queue Actions**버튼을 눌러줍니다.
3. 나오는 메뉴 중에서 **Send Message**를 선택합니다.
4. 나오는 텍스트 상자에 아래의 JSON을 넣어줍니다. (특수 기호 때문에 붙여 넣기보다는 직접 입력을 추천합니다. 대소문자 구분에 주의하십시오. 정상 동작하지 않을 경우, CloudWatch Logs에 남아 있는 lambda 출력을 확인하세요.)
{ "PlayerName" : "SomeName" , "WinDiff" : 1, "LoseDiff" : 0, "ScoreDiff" : 100 }
5. 그리고 **Send Message**버튼을 눌러줍니다.
6. 이제 DynamoDB콘솔로 옮겨가서 <https://console.aws.amazon.com/dynamodb> **GomokuPlayerInfo** 테이블을 열어줍니다. **Items** 탭을 보면 위에서 넣어준 JSON데이터에 해당하는 항목이 반영되어 있는 것을 확인 할 수 있습니다. (이미 같은 PlayerName에 해당하는 항목이 있다면 Win이 1증가하고 Score가 100증가했을 것입니다.)

Section 3: Face the users with API Gateway and S3

우리의 Lambda 함수 중 한가지가 API Gateway를 필요로 한다는 것을 알고 있습니다. 이것은 개발 중의 애플리케이션의 Lambda 함수를 실행할 일종의 "Gateway" 역할을 하기 때문에 API Gateway라 이름이 붙여졌습니다.

1. 우선 콘솔에서 API Gateway 메뉴로 이동합니다.

<https://console.aws.amazon.com/apigateway>

2. Create API를 선택하고, API name은 gomoku을 입력한 뒤 Create API 버튼을 클릭합니다.

Choose the protocol

Select whether you would like to create a REST API or a WebSocket API.

REST WebSocket

Create new API

In Amazon API Gateway, a REST API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

New API Clone from existing API
 Import from Swagger or Open API 3 Example API

Settings

Choose a friendly name and description for your API.

API name*

gomoku

Description

Endpoint Type

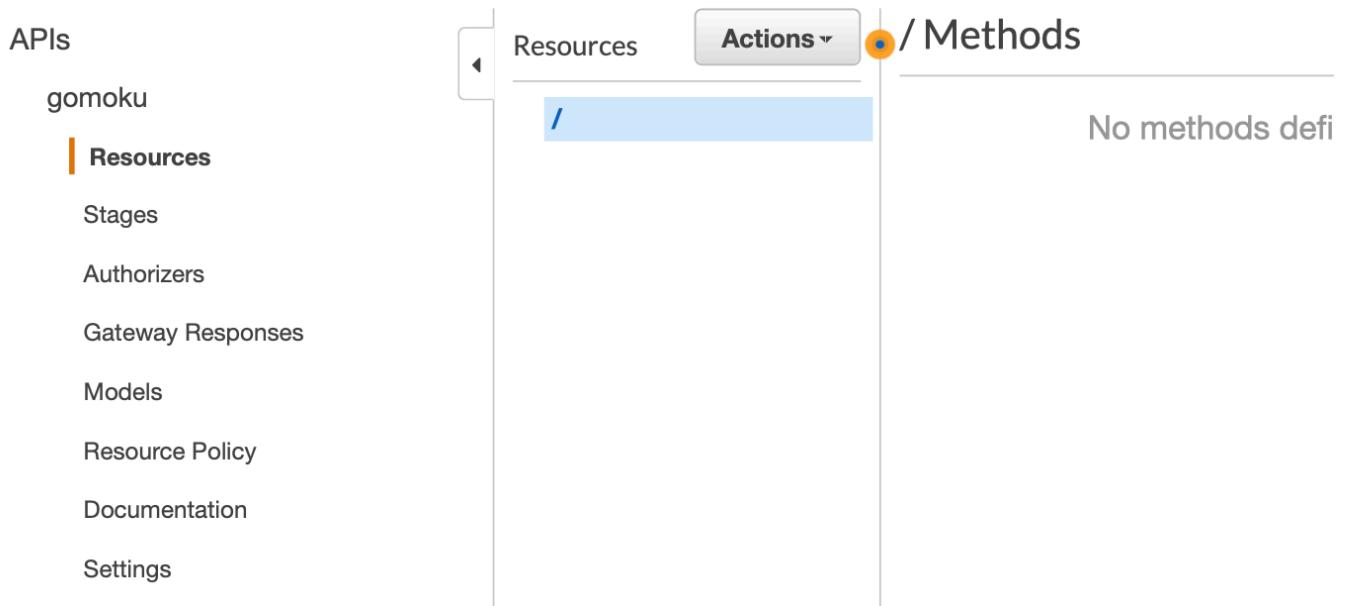
Regional



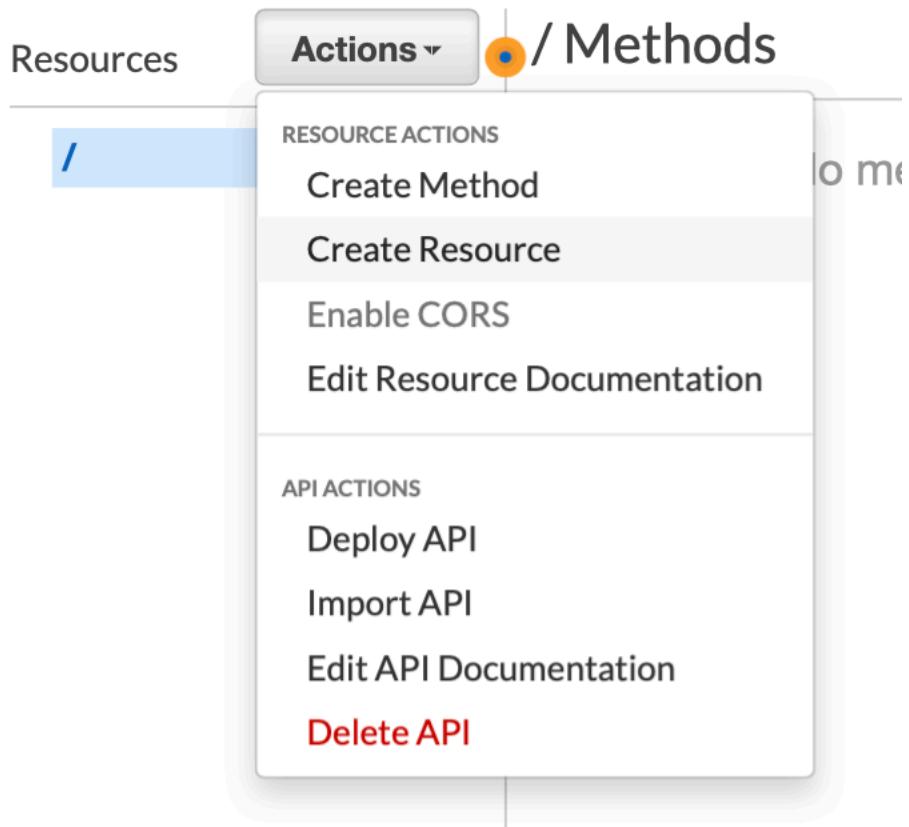
* Required

Create API

3. API가 생성된 뒤에는 다음과 같은 빈 화면이 보일 것입니다.



4. 우선 Resource를 만들고 그 Resource에 Method를 생성합니다. **Actions** 버튼을 클릭하고 resource를 만듭니다.



5. 아래와 같이 Resource Name에 ranking을 입력하고 Enable API Gateway CORS를 선택합니다. Create Resource를 클릭합니다.

New Child Resource

Use this page to create a new child resource for your resource.

Configure as proxy resource



Resource Name*

ranking

Resource Path*

/ ranking

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring **/{proxy+}** as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to **/foo**. To handle requests to **/**, add a new ANY method on the **/** resource.

Enable API Gateway CORS

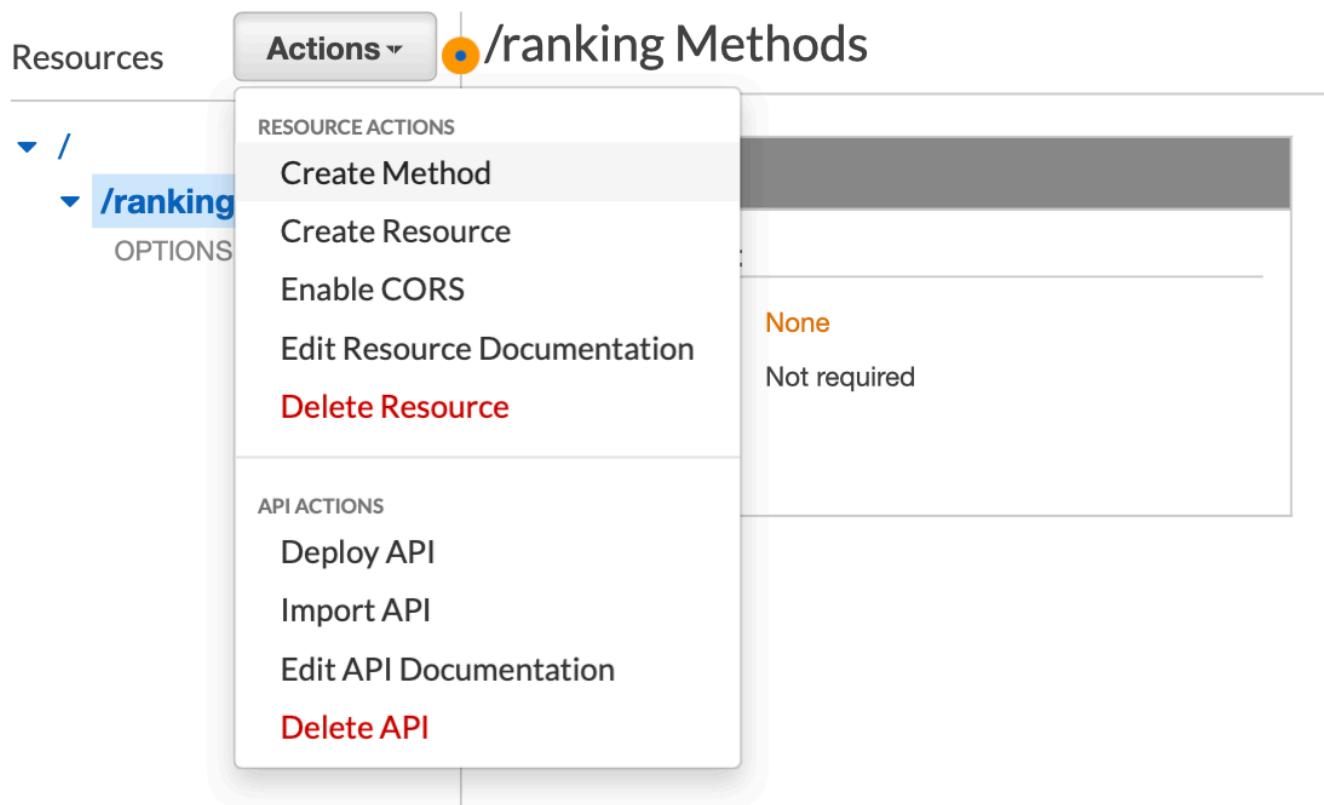


* Required

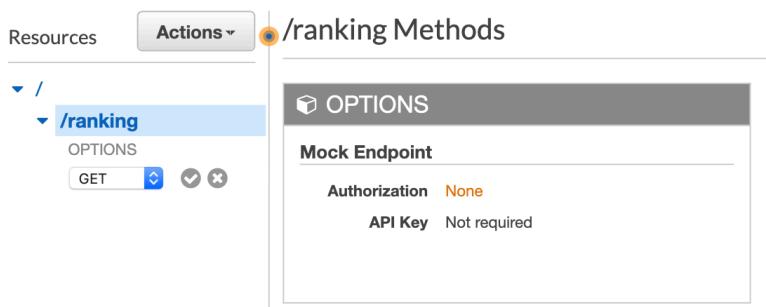
Cancel

Create Resource

6. 다음은 새로운 Method를 생성하는 것입니다. Actions 버튼을 클릭한 뒤 **Create Method** 메뉴를 선택합니다.



7. 아래 작은 리스트박스가 보일 것입니다. GET을 선택한 뒤 옆의 체크 버튼을 클릭합니다.



8. GET 메소드의 상세 설정에서 Integration type은 **Lambda Function**을 선택하고 Lambda Region에 실습을 진행 중인 Region을 선택합니다. Lambda Function에는 앞서 생성한 **game-rank-reader**를 선택한 뒤 **Save** 버튼을 클릭합니다.

The screenshot shows the AWS Lambda API configuration interface. At the top, it says '/ranking - GET - Setup'. Below that, a note says 'Choose the integration point for your new method.' A radio button is selected for 'Lambda Function'. Other options like 'HTTP', 'Mock', 'AWS Service', and 'VPC Link' are available but not selected. A checkbox for 'Use Lambda Proxy integration' is unchecked. The 'Lambda Region' dropdown is set to 'ap-northeast-2'. Under 'Lambda Function', the name 'game-rank-reader' is typed into a text input field. A tooltip icon is shown next to the input field. A checkbox for 'Use Default Timeout' is checked. At the bottom right is a blue 'Save' button.

9. API 구성이 되었습니다. 이제 prod 단계에 배포를 해보겠습니다. **Actions** 버튼을 클릭하고 **Deploy API** 메뉴를 클릭합니다.

The dialog box is titled 'Deploy API'. It contains instructions: 'Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.' There are four input fields: 'Deployment stage' (set to '[New Stage]'), 'Stage name*' (empty), 'Stage description' (empty), and 'Deployment description' (empty). At the bottom are 'Cancel' and 'Deploy' buttons.

10. [New Stage] 를 선택하고 Stage name에는 **prod**를 입력합니다. **Deploy** 버튼을 클릭하여 진행합니다.
11. 완료되면 다음 스크린 캡처와 같이 Stage 구성이 된 것을 확인할 수 있습니다. Prod 배포의 /ranking 메뉴 하단의 GET을 선택하면 나오는 Invoke URL을 기록해둡니다. 이

후 S3를 이용한 정적 웹페이지 구성에 사용됩니다.

Stages **Create**

prod - GET - /ranking

Invoke URL: <https://qzlwjajase.execute-api.ap-northeast-2.amazonaws.com/prod/ranking>

Use this page to override the **prod stage** settings for the GET to /ranking method.

Settings Inherit from stage
 Override for this method

이제 S3가 호스팅하는 웹 사이트를 생성할 것입니다. 버킷에 Ranking board html파일과 Javascript 파일을 업로드하는 것만으로 쉽게 웹 사이트를 호스팅할 수 있습니다.

1. S3 서비스로 이동하여 웹 사이트에 사용할 S3 버킷을 생성합니다. 버킷 생성 시, public object를 허용하기 위해 Block all public access를 uncheck합니다.

Full Stack Game Demo Serverless Full Stack with Gomoku

Note: You can grant access to specific users after you create the bucket.

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, or both. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block *all* public access. These settings apply only to this bucket. AWS recommends that you turn on Block *all* public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block all public access

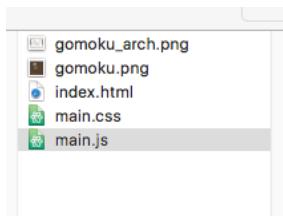
Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

- Block public access to buckets and objects granted through *new* access control lists (ACLS)**
S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects.
This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
- Block public access to buckets and objects granted through *any* access control lists (ACLS)**
S3 will ignore all ACLs that grant public access to buckets and objects.
- Block public access to buckets and objects granted through *new* public bucket policies**
S3 will block new bucket policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
- Block public and cross-account access to buckets and objects through *any* public bucket policies**
S3 will ignore public and cross-account access for buckets with policies that grant public access to buckets and objects.

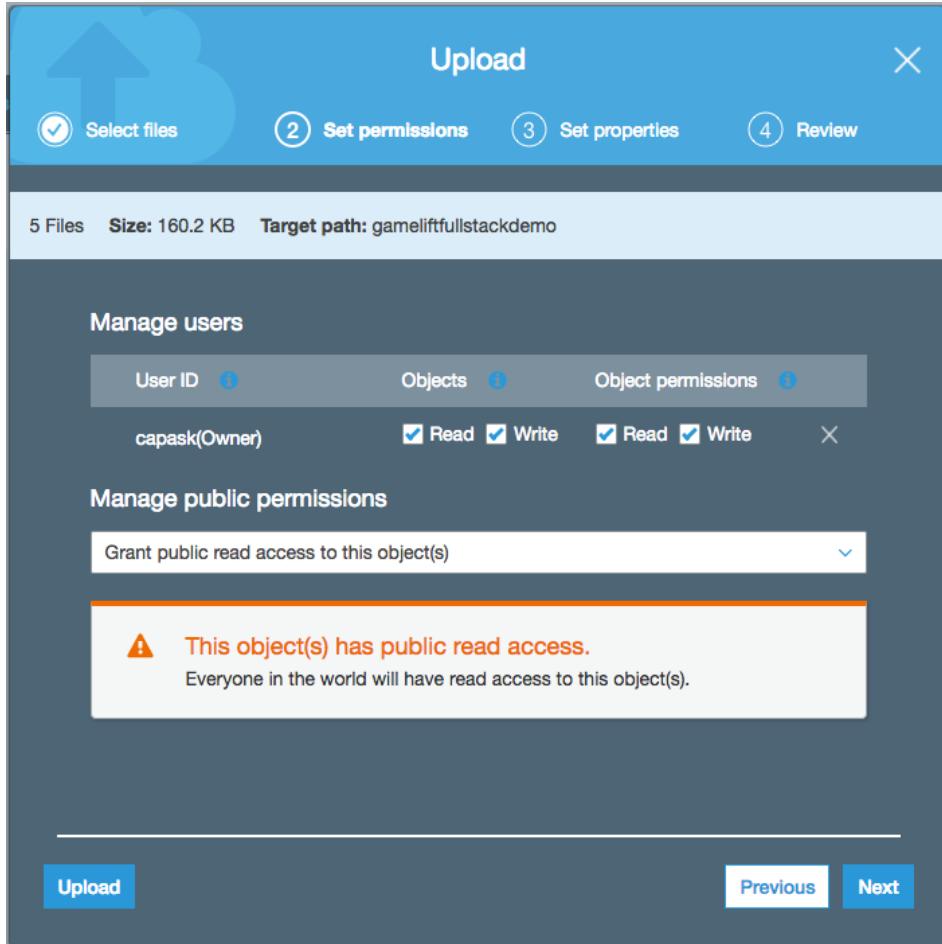
Manage system permissions

[Previous](#) [Next](#)

2. 실습에 사용되는 파일 중 web 디렉토리가 있을 것입니다. 디렉토리 내의 **main.js** 파일을 텍스트 편집기로 엽니다.
3. 48번째 줄의 API Endpoint에 위에서 생성한 API Gateway의 Invoke URL로 수정합니다.
4. 수정한 main.js를 저장한 뒤 web 디렉토리의 파일을 전부 앞서 생성한 버킷에 업로드해줍니다.



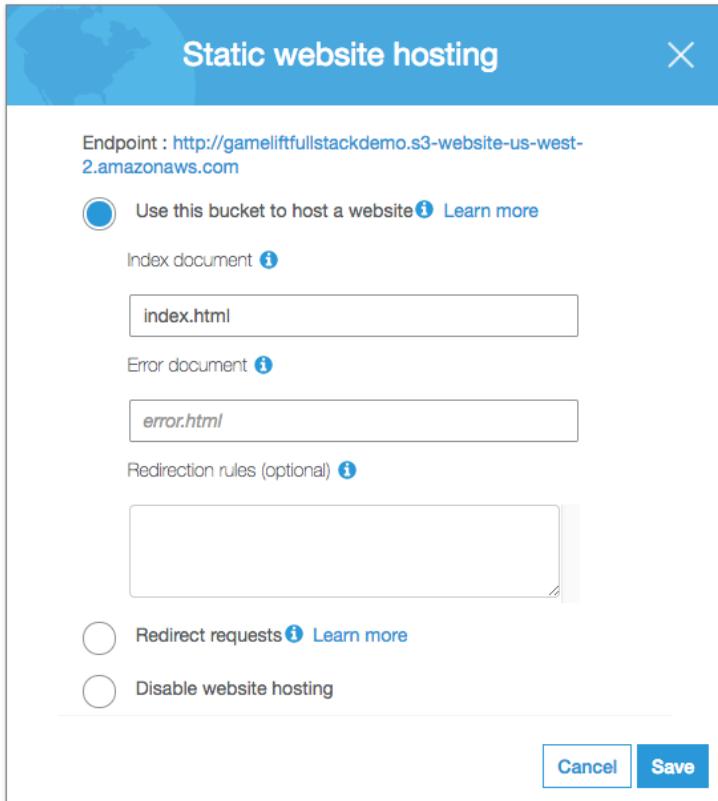
5. 업로드할 때 Public read access 권한을 부여합니다.



6. 버킷에서 Static website hosting을 활성화해주어야 합니다. 버킷의 **Properties** 탭으로 이동합니다.

The screenshot shows the 'Properties' tab of an AWS S3 bucket. It includes tabs for Overview, Properties, Permissions, and Management. Under the Properties tab, there are sections for Versioning, Logging, and Static website hosting. The Static website hosting section is currently disabled. Buttons at the bottom include 'Learn more' and 'Disabled'.

7. Static website hosting을 활성화합니다. Index document에는 **index.html**을 입력한 뒤 **Save** 버튼을 클릭합니다.



8. 파일 업로드를 완료하면 Static website hosting의 Endpoint로 접속하여 단순한 웹 페이지를 확인할 수 있습니다.
9. API Gateway로 돌아와서 CORS 설정을 해주어야 합니다. 이를 통해 Ranking board 정보를 웹 페이지에서 읽어 올 수 있게 됩니다.
10. API Gateway로 돌아와서 **Actions**버튼을 클릭하고 **Enable CORS** 옵션을 선택합니다.
11. CORS 페이지에서 Access-Control-Allow-Origin에 static website URL로 수정해줍니다.
(URL 뒤의 / 가 영향을 미칠 수 있기 때문에 확실하지 않으면 기본 값인 *로 진행합니다.)

The screenshot shows the AWS Lambda API Gateway interface. On the left, there's a tree view of resources under the path '/ranking'. Under this path, 'GET' and 'OPTIONS' methods are listed. In the center, the 'Actions' dropdown is open, and the 'Enable CORS' option is selected. This leads to a configuration page for 'Gateway Responses for gomoku API'. It shows 'Methods' with 'GET' and 'OPTIONS' checked. Below that are three fields: 'Access-Control-Allow-Methods' set to 'GET, OPTIONS', 'Access-Control-Allow-Headers' containing 'Content-Type,X-Amz-Date,Authorizatio', and 'Access-Control-Allow-Origin*' which is empty. At the bottom right of this page is a blue button labeled 'Enable CORS and replace existing CORS headers'.

12. Enable CORS 버튼을 클릭하여 진행합니다.

13. 정상적으로 완료되었다면 왼쪽 탭에 OPTIONS 가 추가된 것을 확인할 수 있습니다.

This screenshot shows the same AWS Lambda API Gateway interface after the CORS configuration has been applied. The left sidebar shows the '/ranking' resource with both 'GET' and 'OPTIONS' methods listed. The central area displays a list of successful steps: 'Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Method Response Headers to OPTIONS method', 'Add Access-Control-Allow-Headers, Access-Control-Allow-Methods, Access-Control-Allow-Origin Integration Response Header Mappings to OPTIONS method', 'Add Access-Control-Allow-Origin Method Response Header to GET method', and 'Add Access-Control-Allow-Origin Integration Response Header Mapping to GET method'. Below this list, a message states: 'Your resource has been configured for CORS. If you see any errors in the resulting output above please check the error message and if necessary attempt to execute the failed step manually via the Method Editor.'

14. 마지막으로 Actions 메뉴의 Deploy API 버튼을 클릭하여 prod 단계에 배포합니다.

배포한 Static Page를 웹브라우저에서 열어주거나 10번에서 기록한 API 의 Invoke URL을 브라우저에서 열어주시면 Section 2에서 테스트로 입력했던 데이터가 표시되거나 JSON(직접 Invoke URL을 열었을 경우) 으로 표기 되는 것을 확인 할 수 있습니다.

Section 4: Putting all together in GameLift

이제 서버 바이너리를 Gamelift 서비스와 함께 동작하도록 구성하는 작업을 시작하겠습니다. 서버 바이너리는 컴파일 된 형태로 제공되었기 때문에 추가 작업이 필요하지는 않습니다. 하지만 직접 컴파일하여 생성한 바이너리를 소스 코드로 사용하실 수도 있습니다.

1. 이번 실습에는 이미 컴파일된 파일이 준비되어 있지만, 직접 컴파일 하고 싶으실 경우 Appendix B를 참조할 수 있습니다.
2. GomokuServer.exe, aws-cpp-sdk-*.dll, config.ini, install.bat, aws-cpp-sdk-gamelift-server.dll 파일들이 Binaries/GomokuServer 폴더에서 확인할 수 있습니다.
3. 텍스트 편집기를 통하여 config.ini 파일을 수정합니다. 파일에는 총 3군데 구성 요소가 있습니다. SQS_REGION은 SQS를 생성한 Region입니다. (예: ap-northeast-2). SQS_ENDPOINT에는 SQS의 Endpoint를 입력합니다.
4. ROLE_ARN은 섹션1에서 마지막으로 만들었던 GameLift Fleet을 위한 ROLE_ARN을 입력합니다. (아래의 스크린샷처럼 따옴표 없이 입력해주세요. Role Arn은 Fleet 내부의 인스턴스에서 얻어 올 수도 있으나 현재 데모용 Gomoku 게임서버에서 구현되어 있지 않으므로 명시적으로 구성해 줍니다.)

```
[config]  
  
# GameResult SQS  
SQS_REGION = ap-northeast-2  
SQS_ENDPOINT = https://sqs.ap-northeast-2.amazonaws.com/592446325190/game-result-queue  
ROLE_ARN = arn:aws:iam::592446325190:role/Gomoku-GameLiftFleetRole
```

5. Gamelift는 업로드의 복잡성 때문에 현재 CLI를 통한 업로드 만을 지원합니다. AWS CLI 환경이 구성되어 있지 않다면 Appendix C를 참고하여 구성합니다.
6. GomokuServer 폴더에서 다음의 GameLift 업로드 명령어를 통해 빌드를 업로드 합니다.
aws gamelift upload-build --name "GomokuServer-Build-1" --build-version "1.0.0" --build-root . --region ap-northeast-2
7. 업로드할 때 실습 Region을 올바르게 설정하였는지 확인해야 합니다.
8. 진행하는 중 콘솔의 GameLift 서비스로 가면 빌드가 업로드 되는 것을 확인할 수 있습니다. <https://console.aws.amazon.com/gamelift>

Full Stack Game Demo Serverless Full Stack with Gomoku

Builds

Use this page to monitor and manage your builds in Amazon GameLift. Each build represents a set of game server binaries and supporting files that have been integrated with the GameLift SDK and uploaded to the GameLift service; once a build is uploaded to GameLift, it is shown here along with its status. Builds in Ready status can be deployed to players by creating a fleet from the build. From this page, you can select a build to delete or to create a new fleet; or choose a build name to view additional details.

Create a new build

See full instructions on preparing your server binaries and creating a build in [Uploading a Build to Amazon GameLift](#). Use the AWS CLI to upload a build. In a command line window, enter the following:

For Windows:

```
aws gamelift upload-build --name <your build name> --build-version <your build number> --build-root <local build path> --operating-system WINDOWS_2012
```

For Linux:

```
aws gamelift upload-build --name <your build name> --build-version <your build number> --build-root <local build path> --operating-system AMAZON_LINUX
```

• The upload may take time to complete depending on your game size and connection speed.

The screenshot shows the 'Builds' section of the Amazon GameLift console. At the top, there are buttons for 'Create fleet from build' and 'Delete build'. Below that is a filter dropdown set to 'Status: All' and a message indicating 'Viewing 1 build(s)'. A table lists the build details: Status (Ready), Name (GomokuServer-Build-1), Build ID (build-06b01eca-5411-4...), Version (1.0.0), OS (Windows 2012), Size (14.95 MB), Date created (2017-08-29 16:33:...), and Fl... (0). The 'Status' column has a dropdown arrow, and the 'Fl...' column has a dropdown arrow.

9. 콘솔의 빌드 페이지에서 방금 업로드한 빌드를 선택합니다. **Create fleet from build** 버튼을 클릭합니다. 이를 통해 게임 서버의 fleet을 생성하게 됩니다.

GomokuServer-Build-1

Use this page to manage a game build. From this page, you can view information about the build, such as the operating system it was built on, and the files it contains. You can also create a new fleet with this build, edit the build, or delete the build.

The screenshot shows the 'Actions' dropdown menu for the build 'GomokuServer-Build-1'. The menu items are 'Create fleet from build' (highlighted in yellow), 'Edit build', and 'Delete build'. The build details table below shows 'Status: Ready', 'Name: GomokuServer-Build-1', and 'Build ID: build-9fb42:'. There is also a 'Fleets' tab at the bottom.

10. 다음의 정보를 입력합니다. 언급이 없는 부분은 기본값으로 진행합니다.

Name: GomokuGameServerFleet-1

Instance Type: C4.large

Fleet type: Spot

Instance Role ARN: 섹션1의 마지막에서 GameLift Fleet을 위해 만들었던 Role의 ARN

Launch path: GomokuServer.exe (직접 컴파일 했다면 컴파일한 바이너리명)

Full Stack Game Demo
Serverless Full Stack with Gomoku

Concurrent processes: 50

우측의 녹색 체크 버튼을 클릭하여 확인합니다.

Process management

Server process allocation

Configure how to launch server processes and specify the number of server processes to run concurrently on each instance. To run server processes with a different executable, or the same executable with different launch parameters, add more configurations.

Launch path*	Launch parameters	Concurrent processes*
C:\game\GomokuServer.exe		50
+ Add configuration Multiple configurations and concurrent processes are supported with game servers using the Amazon GameLift SDK v.3.0+ (included in Lumberyard v.1.4+). Game servers using earlier versions can support only one concurrent process. If the fleet is configured for more than one concurrent process, it will fail to activate.		

Game session activation

Configure the number and frequency of game session activations to allow on each instance. Limiting the volume of game session activations on a single instance can reduce strain on resources and prevent impacting game server performance.

Max concurrent game session activation* No limit Limited to per instance

New activation timeout* 600 seconds

EC2 Port Settings: Port range; 49152-60000

Protocol; TCP, IP address range; 0.0.0.0/0 입력후 녹색 체크 버튼

EC2 port settings

Set IP address and port ranges to allow inbound access to this fleet. Each server process in this fleet must use an IP address and port in these ranges.

Port range*	Protocol*	IP address range*
49152-60000	TCP	0.0.0.0/0
+ Add port settings		

Initialize fleet를 클릭하고 잠시 기다립니다.

Initialize fleet

11. 생성이 시작되면 Fleet에서 다음과 같은 화면을 확인할 수 있습니다.

Full Stack Game Demo
Serverless Full Stack with Gomoku

The screenshot shows the Amazon GameLift Fleet management interface. At the top, there's a navigation bar with tabs for 'Amazon GameLift' and 'Fleets'. The 'Fleets' tab is selected, and the sub-tab 'GomokuGameServerFleet-1' is shown. A search bar and a help icon are also present. Below the navigation, the title 'Fleet: GomokuGameServerFleet-1' is displayed, followed by a subtitle: 'A fleet manages one build of your game server, which is replicated across many Amazon EC2 instances. Use this page to monitor your fleet.' A table provides detailed information about the fleet:

Status	Fleet ID	EC2 type	OS	Active instances	Active servers	Game sessions	Player sessions
New	fleet-8f1ddd6e-d2dc-43ad-83d9-7ad9c9d5b828	c3.large	Windows 2012 R2	0	0	0	0 of 0

12. 생성이 완료되었다면 왼쪽의 파란 박스가 Active 상태의 녹색으로 변할 것입니다.

시간이 약 20분 정도 소요됩니다.

13. Fleet을 생성하는 동안에 Alias를 생성하겠습니다.

14. 메뉴에서 **Create alias** 옵션을 선택합니다. 그리고 Alias name과 Description을 입력해줍니다.

15. Routing options의 Type은 **Simple**을 선택하고, Associated fleet에서 **[Select fleet]**을 선택한 뒤 생성한 fleet을 선택합니다.

Create alias



Alias details

Provide a name and description for your alias.

Alias name*	<input type="text" value="GomokuAlias"/>
Description	<input type="text" value="Prod"/>

Routing options

Set up your simple alias to resolve and route traffic to a specific fleet. If you designate an alias as terminal, traffic that tries to connect to that alias will receive a terminal exception along with a string that you define.

Type*	<input type="button" value="Simple"/>
Associated fleet*	<input type="text" value="GomokuGameServerFleet-1"/>

[Cancel](#) [Configure alias](#)

16. Fleet 상태는 진행 상태에 따라 Downloading/ Validating/ Activating 가 있습니다.

Success! You have successfully created alias GomokuAlias.

×

GomokuAlias



Grant permissions to specific Cognito IDs to access your fleet.

[View in dashboard](#)

Actions ▾

Type	Alias ID	ARN
Simple	alias-783adaac-8105-4c26-bbd7-a40d9b90e00e	arn:aws:gamelift:us-west-2::alias/alias-783adaac-8105-4c26-bbd7-a40d9b90e00e

Alias parameters

Fleet ID	Fleet name	Game sessions	Player sessions	Date created
fleet-8f1ddd6e-d2dc-43ad-83d9-7ad9c9d5b828	GomokuGameServerFleet-1	0	0	2017-08-29 16:44:39 UTC

17. 실제로 Gamelift fleet을 이용하기 위하여 Alias ID를 사용하게 됩니다. 해당 Alias ID를 기록해둡니다.

기본적으로 240분 동안 아무 활동이 없다면 Fleet은 인스턴스 0개로 스케일-인 합니다. 이번 실습을 진행하는 동안에는 문제가 없지만 조금 더 오래 실행하고 싶다면 auto scale parameter를 최소한 1로 변경해야 합니다. 아니면 직접 1개의 인스턴스를 실행하도록 override해주어야 합니다.

Section 5: Serverless FlexMatch 구성하기

이제 FlexMatch 를 구현하기 위한 GameLift, Lambda, API Gateway 를 구성합니다.

FlexMatch 를 사용하기 위해서는 GameLift 의 Queue 서비스를 사용해야 합니다.

1. 메뉴에서 **Create a queue** 를 선택합니다.
2. 아래와 같이 Queue 의 이름을 입력하고, Add destination 에서 앞서 생성한 Alias 를 선택합니다. **Create queue** 를 선택하여 queue 를 생성합니다.

Create queue

Queue details

Configure a queue to process requests for new game sessions. Add destinations from any region to host new game sessions for this queue, and list them in order of preference. Add player latency policies to protect players from very high latencies.

Queue Name*	gomoku-queue	
Queue Timeout*	600	

Player latency policies

Add a policy to set a maximum acceptable player latency for new game sessions, and specify the amount of time to enforce the policy. Multiple policies are automatically reordered by latency maximum, with the lowest value listed first.

[Add player latency policy](#)

Destinations

Priority (default)	Region*	Type*	Fleet or Alias name*		
1	ap-northeast-1	Alias	alias-20170517101352-192-192-gamelift-alias		

[Add destination](#)

[Cancel](#) [Create queue](#)

3. 다음 단계를 FlexMatch 를 위한 rule 을 만듭니다. **Create matchmaking rule set** 을 선택합니다.
4. 아래와 같이 **Rule set** 의 이름을 입력하고 Rule set 을 넣어 줍니다. Rule set 은 앞서 받은 파일에서 **GomokuRuleSet.json** 의 파일 내용을 복제하여 넣어 줍니다. 여기서의 rule 은 간단하게 score 점수가 300 점 이내의 사용자간에 Match 를 시켜주며 해당 사용자 match 가 시간내에 안 될 경우, rule 을 완화하여 match 시켜 줍니다. Validate rule set 을 선택하여 rule set 이 정상인지 확인 한 후에 create rule set 으로 rule set 을 생성합니다.

Create matchmaking rule set



Rule set details

To create a rule set, assign a name and enter valid rules syntax.

Rule set name* ?

Rule set* ?

Cancel Validate rule set Create rule set

5. 다음은 앞서 만든 Queue 와 Matchmaking rule set 을 연결 시켜 주는 작업을 수행합니다. **Create matchmaking configuration** 을 메뉴에서 선택합니다. 아래와 같이 앞서 만든 queue 와 rule set 을 선택하여 configuration 을 생성합니다.

Create matchmaking configuration



Matchmaking configuration details

To create a matchmaking configuration, assign a name, specify a rule set for matching players and a queue to start a game session for the match. Set additional configuration parameters as needed.

Name* i

Description i

Queue* gomoku-queue i

Request timeout* i

Acceptance required* No Yes, with a timeout of seconds i

Rule set name* i

Notification target i

Additional players i

Custom event data i

Game session data i

Game properties

Key*	Value*
No game properties	

i
[+ Add game properties](#)

Backfill mode* i

[Cancel](#)

[Create](#)

이제 GameLift 의 FlexMatch 설정은 완료하였습니다. 다음은 game client 가 FlexMatch 요청을 하는 Lambda 와 API Gateway 를 구성합니다.

이번 랙에서는 총 2개의 Lambda 함수를 생성할 것입니다. 하나는 client로부터 MatchMaking 요청을 처리하는 람다함수이고, 다른 하나는 MatchMaking 결과를 확인하기 위한 요청을 처리하는 람다입니다. 첫번째로 Matchmaking 요청을 처리하는 람다를 생성합니다.

1. 콘솔에서 **Lambda** 메뉴로 이동합니다. <https://console.aws.amazon.com/lambda>
2. **Create function** 버튼을 클릭하여 첫번째 함수 생성을 시작합니다.
3. **Author from scratch** 메뉴를 선택하여 빈 함수를 우선 생성합니다.
4. **Name** 항목에는 **game-match-request**를 입력합니다.
5. **Runtime**은 Python 2.7을 선택합니다.
6. **Permissions** 항목에서 **Role**은 **Use an existing role**을 선택하고 기존에 만들어둔 **Gomok-game-match-request**를 선택하고 **Create function**을 실행합니다.

Create function Info

Choose one of the following options to create your function.

Author from scratch

Start with a simple Hello World example.



Use a blueprint

Build a Lambda application from sample code and configuration presets for common use cases.



Browse serverless app repository

Deploy a sample Lambda application from the AWS Serverless Application Repository.



Basic information

Function name

Enter a name that describes the purpose of your function.

game-match-request

Use only letters, numbers, hyphens, or underscores with no spaces.

Runtime Info

Choose the language to use to write your function.

Python 2.7

Permissions Info

Lambda will create an execution role with permission to upload logs to Amazon CloudWatch Logs. You can configure and modify permissions further when you add triggers.

▼ Choose or create an execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

Use an existing role

Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

Gomok-game-match-request



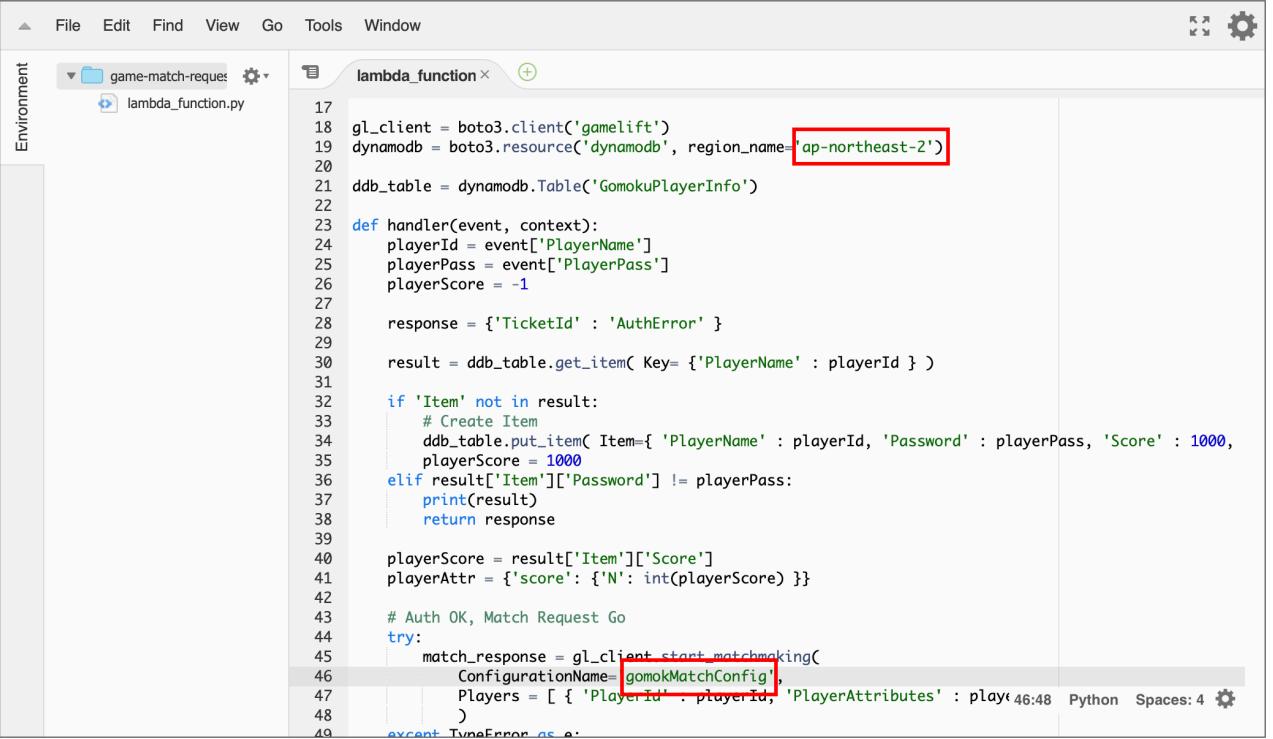
[View the Gomok-game-match-request role](#) on the IAM console.

7. 제공된 소스 파일 중 Lambda 폴더 밑에 있는 MatchRequest.py의 내용을 Lambda 코드창에 복제해 넣습니다. 코드 상의 Region과 Match Config의 이름이 앞서 생성한 이름과 동일한지 확인합니다. (대소문자 일치하도록 확인)

Full Stack Game Demo Serverless Full Stack with Gomoku

Function code [Info](#)

Code entry type Edit code inline	Runtime Python 2.7	Handler Info lambda_function.lambda_handler
---	-----------------------	--



```
17 gl_client = boto3.client('gamelift')
18 dynamodb = boto3.resource('dynamodb', region_name='ap-northeast-2')
19
20 ddb_table = dynamodb.Table('GomokuPlayerInfo')
21
22 def handler(event, context):
23     playerId = event['PlayerName']
24     playerPass = event['PlayerPass']
25     playerScore = -1
26
27     response = {'TicketId' : 'AuthError'}
28
29     result = ddb_table.get_item( Key= { 'PlayerName' : playerId } )
30
31     if 'Item' not in result:
32         # Create Item
33         ddb_table.put_item( Item={ 'PlayerName' : playerId, 'Password' : playerPass, 'Score' : 1000,
34         playerScore = 1000
35         elif result['Item']['Password'] != playerPass:
36             print(result)
37             return response
38
39         playerScore = result['Item']['Score']
40         playerAttr = { 'score': { 'N': int(playerScore) } }
41
42         # Auth OK, Match Request Go
43         try:
44             match_response = gl_client.start_matchmaking(
45                 ConfigurationName='gomokMatchConfig',
46                 Players = [ { 'PlayerId' : playerId, 'PlayerAttributes' : play
46:48 Python Spaces: 4
47             )
48         except TypeError as e:
49             print(e)
```

8. Lambda 함수의 실행 시간 및 메모리 설정을 아래와 같이 설정합니다.

Basic settings

Description

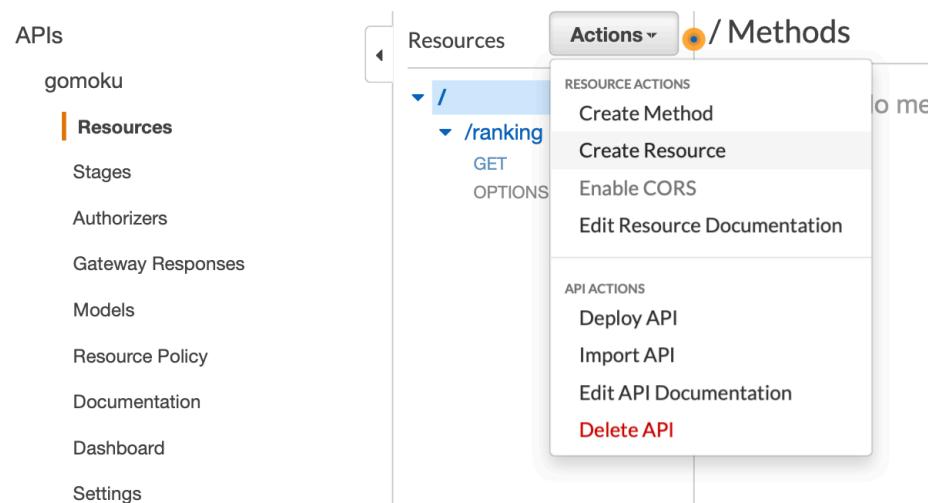
Memory (MB) [Info](#)
Your function is allocated CPU proportional to the memory configured.
 128 MB

Timeout [Info](#)
1 min 0 sec

9. **Save** 버튼을 클릭하여 함수를 생성합니다. 생성한 함수는 game client 의 Match making 요청을 받아 DynamoDB 에서 사용자의 정보를 읽고 이에 기반하여 GameLift 에 match making 을 요청합니다.
10. **Create function** 버튼을 클릭하여 두번째 함수 생성을 시작합니다.
11. **Author from scratch** 메뉴를 선택하여 빈 함수를 우선 생성합니다.
12. **Name** 항목에는 **game-match-status**를 입력합니다.
13. **Runtime**은 Python 2.7을 선택합니다.
14. **Permissions** 항목에서 **Role** 은 Use an existing role을 선택하고 기존에 만들어둔 **Gomok-game-match-status**를 선택하고 Create function을 실행합니다.
15. 제공된 소스 파일 중 Lambda 폴더 밑에 있는 MatchStatus.py 의 내용을 Lambda 코드창에 복제해 넣습니다.
16. 앞서와 같이 Lambda 함수 실행시간을 1 분으로 조정합니다.
17. **Save** 버튼을 클릭하여 함수를 생성합니다. 생성한 함수는 game client 가 TicketId 로 자신이 요청한 match making 요청이 완료되었는지를 확인하고 요청이 완료되었다면 client 가 접근할 게임서버의 IP 주소와 Port 를 알려 줍니다.

다음은 앞서 생성한 Lambda 함수를 game client 가 호출할 수 있도록 API Gateway 를 설정합니다.

1. 우선 콘솔에서 API Gateway 메뉴로 이동합니다.
<https://console.aws.amazon.com/apigateway>
2. 앞서 생성한 gomoku API를 선택하고 Create Resource를 선택합니다.



3. Resource name은 matchrequest를 입력하고 Enable API Gateway CORS를 체크하고 Create Resource 버튼을 클릭하여 Resource를 생성합니다.

New Child Resource

Use this page to create a new child resource for your resource. 

Configure as  proxy resource 

Resource Name* 

matchrequest

Resource Path* 

/ matchrequest

You can add path parameters using brackets. For example, the resource path **{username}** represents a path parameter called 'username'. Configuring /{proxy+} as a proxy resource catches all requests to its sub-resources. For example, it works for a GET request to /foo. To handle requests to /, add a new ANY method on the / resource.

Enable API Gateway CORS 

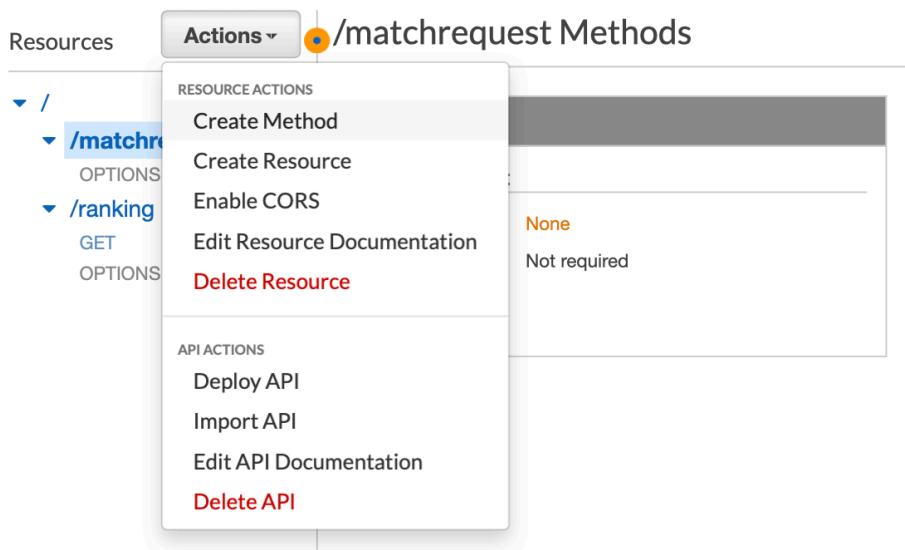


* Required

[Cancel](#)

[Create Resource](#)

4. 생성한 Resource에 새로운 Method를 생성합니다. Actions 버튼을 클릭한 뒤 Create Method 메뉴를 선택합니다.



The screenshot shows the AWS Lambda console interface. On the left, there's a tree view of resources under the path '/'. Under '/matchrequest', there are 'OPTIONS' and 'GET' methods. A context menu is open over the '/matchrequest' node, with 'Actions' selected. The 'Actions' dropdown menu has two sections: 'RESOURCE ACTIONS' and 'API ACTIONS'. In the 'RESOURCE ACTIONS' section, 'Create Method' is highlighted. A sub-menu for 'Create Method' is open, showing options: 'None' (selected) and 'Not required'. In the 'API ACTIONS' section, 'Delete API' is highlighted.

5. 아래 작은 리스트박스가 보일 것입니다. **POST**를 선택한 뒤 옆의 체크 버튼을 클릭합니다.

The screenshot shows the AWS Lambda API configuration interface. On the left, there's a tree view of resources under a root path. Under the /matchrequest resource, the POST method is selected. In the main panel, there's a 'OPTIONS' method configuration section. It shows 'Mock Endpoint' settings: 'Authorization' is set to 'None' and 'API Key' is marked as 'Not required'. There are also dropdown menus for 'HTTP Method' (set to 'POST') and 'Auth Type' (with a checked checkbox).

6. POST 메소드의 상세 설정에서 Integration type은 **Lambda Function**을 선택하고 Lambda Region에 실습을 진행 중인 Region을 선택합니다. Lambda Function에는 앞서 생성한 **game-match-request**를 선택한 뒤 **Save** 버튼을 클릭합니다.

The screenshot shows the 'POST - Setup' configuration page for the /matchrequest API. The 'Integration type' dropdown is set to 'Lambda Function'. Below it, there are other options: 'HTTP', 'Mock', 'AWS Service', and 'VPC Link'. A checkbox for 'Use Lambda Proxy integration' is unchecked. The 'Lambda Region' dropdown is set to 'ap-northeast-2'. In the 'Lambda Function' dropdown, 'game-match-request' is selected. At the bottom right, there's a 'Save' button.

7. 두 번째 람다함수를 연동하기 위해 생성한 API에 두 번째 Resource를 만듭니다. **API 의 root**를 선택하고 Actions에서 Create Resource를 선택합니다.

The screenshot shows the AWS API Gateway console. On the left, there's a tree view of resources under the path `/matchre`. The `POST` method is selected. A context menu is open over this method, with the title `/ Methods`. The menu is divided into two sections: `RESOURCE ACTIONS` and `API ACTIONS`. In the `RESOURCE ACTIONS` section, `Create Method` is at the top, followed by `Create Resource` (which is highlighted), `Enable CORS`, and `Edit Resource Documentation`. In the `API ACTIONS` section, `Deploy API`, `Import API`, `Edit API Documentation`, and `Delete API` are listed.

8. 앞서 첫번째 Resource 와 동일한 방식으로 생성합니다 **Resource Name** 은 `matchstatus` 로 입력합니다. **Enable API Gateway CORS** 를 체크합니다.

The screenshot shows the `New Child Resource` page. The left sidebar shows the path `/matchrequest` with a `POST` method selected. The main form has the following fields:

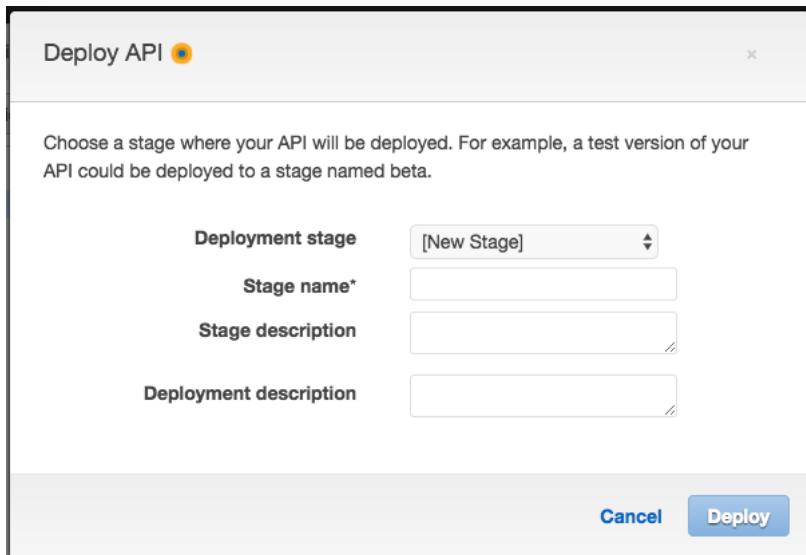
- Configure as proxy resource:** An unchecked checkbox with a help icon.
- Resource Name***: The input field contains `matchstatus`.
- Resource Path***: The input field contains `/ matchstatus`. Below it is a note explaining path parameters using brackets, mentioning `{username}` as an example.
- Enable API Gateway CORS**: A checked checkbox with a help icon.

At the bottom, there are buttons for `* Required`, `Cancel`, and `Create Resource`.

9. 앞서 첫번째 Resource 에 생성한 Method 와 동일한 방식으로 **Create Method** 를 수행합니다.

10. POST 메소드를 생성하고 **game-match-status** 램다를 연동하고 Save 버튼을 클릭하여 저장합니다.

11. API 구성이 되었습니다. 이제 배포를 수행합니다. **Actions** 버튼을 클릭하고 **Deploy API** 메뉴를 클릭합니다.



12. [New Stage]를 선택하고 Stage name에는 prod를 입력합니다. Deploy 버튼을 클릭하여 진행합니다.

13. 완료되면 API의 Invoke URL이 표시됩니다. 아래 URL은 game client 설정에 사용됩니다.

prod Stage Editor

Delete Stage Configure Tags

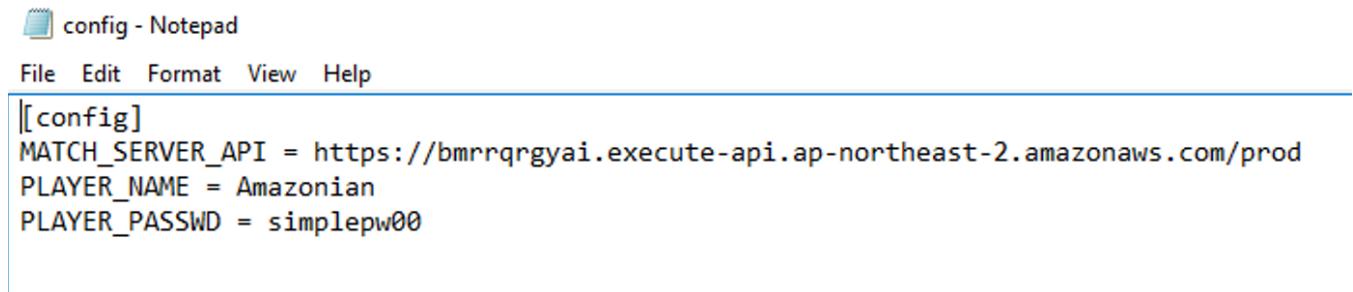
Invoke URL: <https://bmrrqrgyai.execute-api.ap-northeast-2.amazonaws.com/prod>

Section 6: Let's play the client

게임 클라이언트는 기본적으로 Windows OS에서 동작하도록 개발되었습니다. (Mac에서 실행할 수 있는 버전은 따로 제공됩니다. Mac 버전에서 사용을 위해서는 Xquartz를 미리 설치해야 합니다.)

게임 클라이언트는 Outbound TCP 커넥션을 2개 맺습니다. (방화벽 설정이 필요하다면 설정해줍니다.)

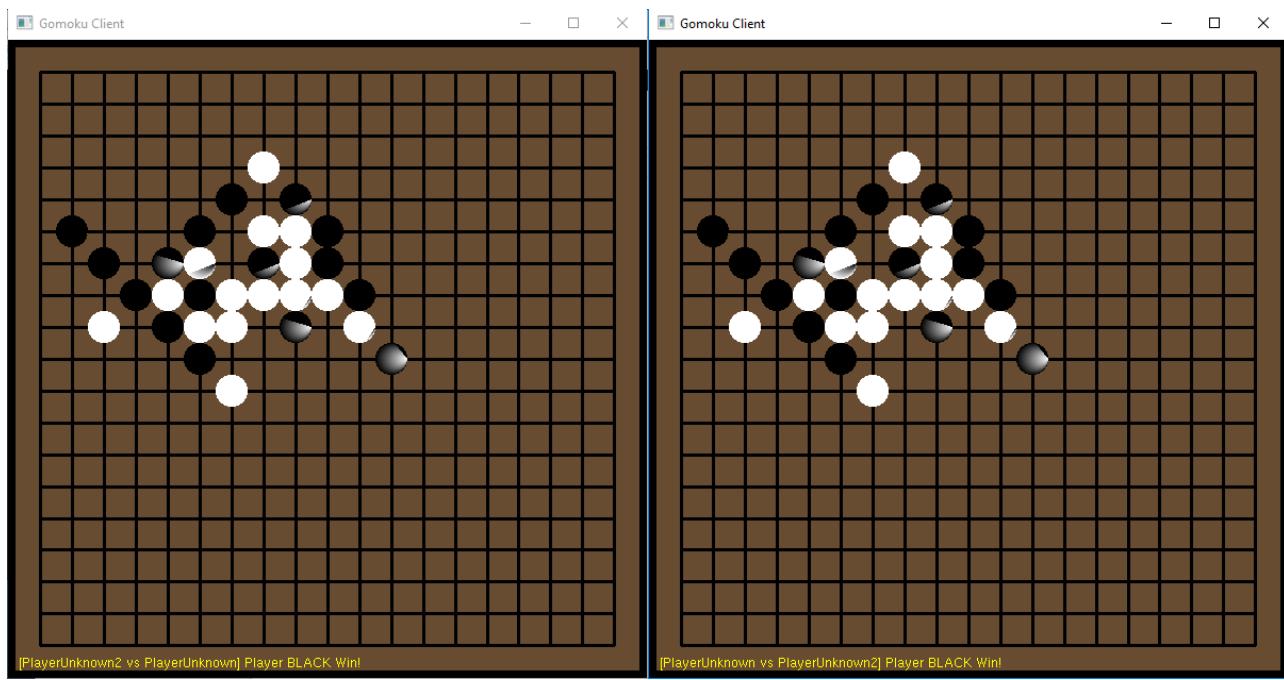
1. 다운로드 받은 실습 파일에서 Binaries/client_player1 폴더로 이동합니다.
2. 해당 client_player1 폴더 내의 config.ini 파일의 수정이 필요합니다.
3. 앞선 Section 5의 마지막 단계에서 확인한 API Gateway URL을 MATCH_SERVER_API값에 입력합니다.



```
[config]
MATCH_SERVER_API = https://bmrrqrgyai.execute-api.ap-northeast-2.amazonaws.com/prod
PLAYER_NAME = Amazonian
PLAYER_PASSWD = simplepw00
```

4. PLAYER_NAME과 PLAYER_PASSWD를 (임의로) 지정합니다.
5. 게임 클라이언트(GomokuClient.exe)를 실행합니다. 아래의 스크립 캡처와 같이 실행될 것입니다.
6. 다른 플레이어를 실행해야 합니다. 다운로드 받은 실습 파일에서 Binaries/client_player2 폴더로 이동하여 config.ini 파일을 앞서와 같이 수정하고 player2를 실행합니다.
7. 두 개의 게임 화면에서 우클릭한 뒤 Start 하세요!
8. 이미 졌다고 판단된다면 우클릭한 뒤 Give up을 할 수 있습니다.
9. DynamoDB의 Score를 임의로 변경하여 Matchmaking을 테스트해 보십시오.
10. 수고하셨습니다!

Full Stack Game Demo
Serverless Full Stack with Gomoku



참고: MatchMaker는 기본적으로 `PLAYER_NAME`가 이미 DynamoDB에 존재하면 `PLAYER_PASSWD`를 비교합니다. 패스워드가 맞다면 로그인이 자동으로 완료 됩니다. 만일 `PLAYER_NAME`에 해당하는 데이터가 없다면 해당 플레이어를 새로 생성하고 자동으로 로그인까지 하도록 구현되어 있습니다. 자세한 구현은 따로 제공되는 MatchMaker 소스 코드를 참고하세요.

게임 클라이언트는 친구, 가족에게 배포하여 Gomoku(Omok, FiveStones)를 함께 즐기세요.
(**바이너리를 재배포 할 때는 법적인 제한 사항을 확인해야 합니다)

Appendix A Deployment Package with Python cheat sheet

How to create a deployment package for Lambda using python

For detailed information, refer to our web page at <http://docs.aws.amazon.com/lambda/latest/dg/lambda-python-how-to-create-deployment-package.html>

For a quick cheat sheet, as python environment in lambda does not have all the library user can possibly use, it is recommended that you use a customized deployment package for any function that extends beyond built-in AWS SDK.

In our case, we created the project directory, and using pip, installed the necessary library into the project directory, and zipped the directory as is.

From CLI, the sequence of commands should look like this.

```
] mkdir project
] cd project
] vi Scoring.py
] pip install redis -t .
] pip install boto3 -t .
] zip -r LambdaDeploy.zip *
```

For more information on creating a proper deployment package, please refer to the link above.

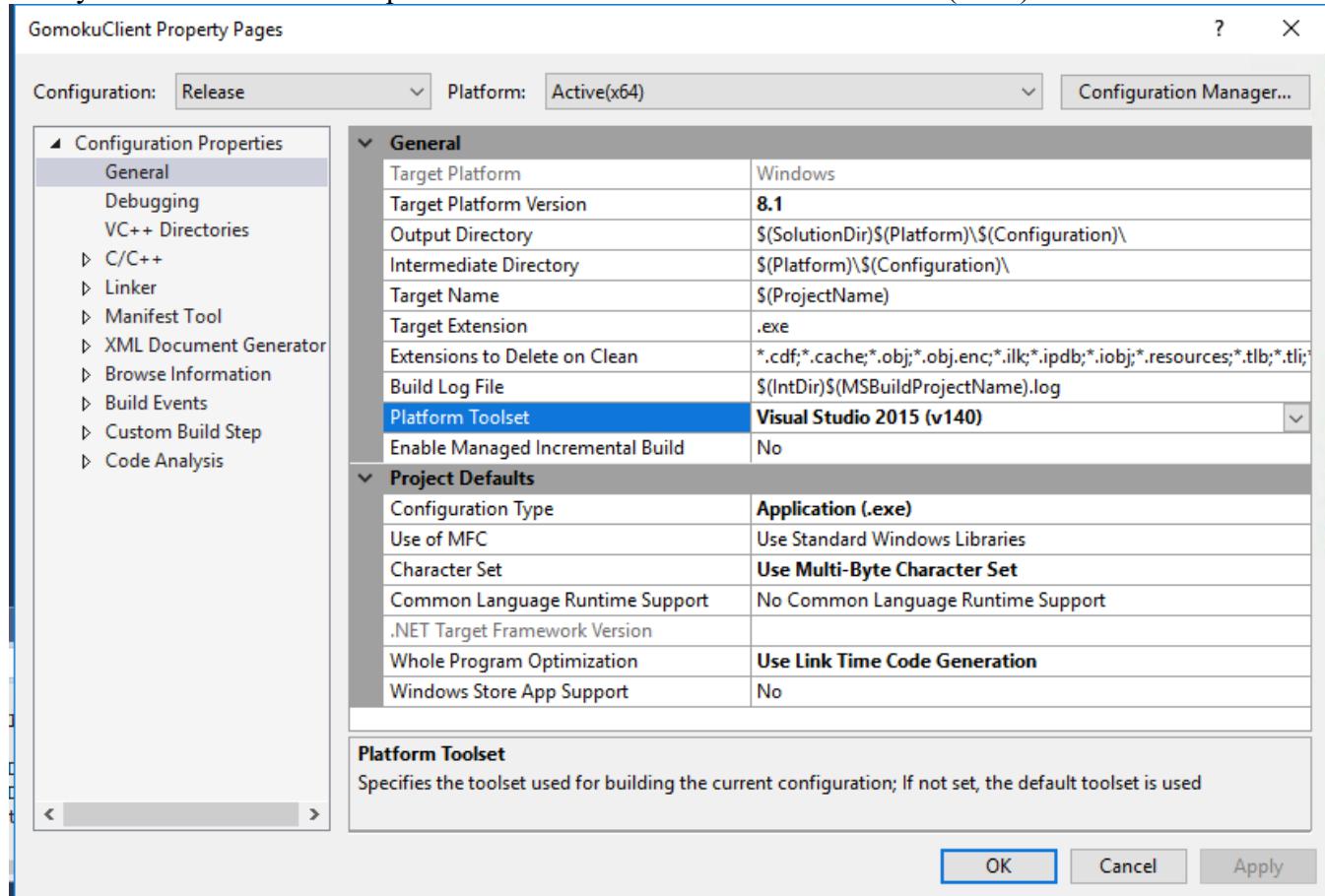
Appendix B Notes on compiling the source binary

Compiling the binaries used in this example.

This lab has 3 different binaries used throughout. 1 Game client, and 2 game server binaries are used. For your convenience, each of them are precompiled and supplied within the starting package, however if you want to compile the binary yourself, here's the simple instruction on compiling.

You need to have Visual Studio 2015 Community Edition installed. If you can reconfigure VS 2017 to match that of 2015, you might try, however due to the NuGet package dependency, it is recommended to use VS 2015 edition instead.

Also you will want to use the platform toolset version Visual Studio 2015 (v140).



The each of the project files, GomokuServer.sln, GomokuMatchMaker.sln, GomokuClient.sln are prepared within the project directory.

As for NuGet, please refer to this link for detailed information on how to update the package manager on Visual Studio 2015.

Appendix C AWS Cli environment notes

Creating a AWS Cli environment.

In this lab, we need to create a cli environment for GameLift binary upload.

Fortunately, we can easily create a AWS CLI environment following below steps.

For installing on each OS, refer to this page:

<http://docs.aws.amazon.com/cli/latest/userguide/installing.html>

Once you have the environment installed, you will need to create a IAM user with programmatic access, and setup your command line interface with the created IAM credentials.

Please refer to this page for detailed instructions:

<http://docs.aws.amazon.com/cli/latest/userguide/cli-chap-getting-started.html>

Appendix D Setting up Windows notes

EC2 Windows Server 생성 및 설정

1. 콘솔에서 EC2 서비스 선택 후 Launch Instance 버튼을 클릭하여 EC2 생성을 시작합니다.
2. Amazon Machine Image 는 Microsoft Windows Server 2016 Base 를 선택합니다.



3. Instance Type 은 t2.large 혹은 더 큰 타입을 선택합니다.
4. 보안 그룹은 다음과 같이 설정합니다.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server an allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

The screenshot shows the 'Create New Security Group' configuration page. It has two radio button options: 'Create a new security group' (selected) and 'Select an existing security group'. Below this, there are fields for 'Security group name' (MatchMaking) and 'Description' (SG for MatchMaking). A table below lists two inbound rules: one for RDP (Protocol TCP, Port Range 3389, Source 0.0.0.0/0) and one for Custom TCP (Protocol TCP, Port Range 5999, Source 0.0.0.0/0). An 'Add Rule' button is visible at the bottom left.

5. 갖고 있는 Key Pair(.pem 파일이 있다면)가 있다면 그대로 사용합니다. Key Pair 가 없는 경우 Create a new key pair 옵션을 선택하여 생성 후 다운로드합니다. 생성한 Key Pair 는 인스턴스의 패스워드 복호화에 사용되기 때문에 패스워드가 지정되지 않은 Key Pair 를 사용해야합니다.
6. Windows Server 의 경우 접속 가능 상태까지 약 4 분이 소요됩니다.
7. EC2 서비스의 Instances 메뉴로 이동하여 생성한 인스턴스를 선택한 뒤 상단의 Connect 버튼을 클릭합니다.
8. 기본 계정은 Administrator 가 할당되며 Get Password 버튼을 클릭하여 접속 패스워드를 얻습니다.
9. 패스워드는 Public Key 를 통해 생성되며 암호화 되어 있습니다.
10. 다음과 같이 인스턴스 생성할 때 사용한 Key Pair 와 동일한 Key 를 선택한 뒤 Decrypt Password 버튼을 클릭하여 패스워드를 복호화합니다.

Connect To Your Instance > Get Password X

The following Key Pair was associated with this instance when it was created.

Key Name virginia.pem

In order to retrieve your password you will need to specify the path of this Key Pair on your local machine:

Key Pair Path virginia.pem

Or you can copy and paste the contents of the Key Pair below:

```
-----BEGIN RSA PRIVATE KEY-----  
MIIEpAIBAAKCAQEAtzITIHAc0zSPuBVzA0m+dMBBy4957TWpjxW5EXYFWwtf0DBuaBPGxCCWCIDwT  
eqjDcXeUOYdRUNDgp8rJ/D6rljP8W8c8fefJkJtddlIaqAsVyP2y8SDkbyl8+8t5AVfQbzjGGELX  
Nq6dRJtGI8q3YIDZ/8TWt5+7WkJWF4v03jjNaionbiaaE6i5Hv/mYmMPh3/Dn7qgMyodpUiLJ0Qu  
jKAzmlZr6KwyBgokrG0WGeWfIYABfgogC7kSJenmCYDKnkjVFjiaPfrz91gPC4FNbZ9HIBK/WzJK
```

Back Close

11. 다음과 같이 인스턴스 접속 정보가 화면에 나타납니다.

Connect To Your Instance X

You can connect to your Windows instance using a remote desktop client of your choice, and by downloading and running the RDP shortcut file below:

When prompted, connect to your instance using the following details:

Public DNS ec2-34-230-70-26.compute-1.amazonaws.com

User name Administrator

Password [REDACTED]

If you've joined your instance to a directory, you can use your directory credentials to connect to your instance.

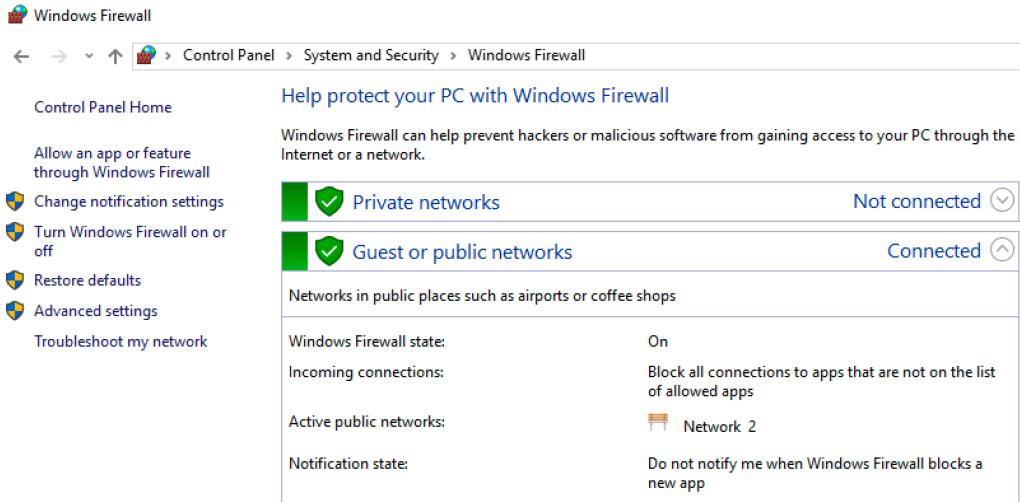
If you need any assistance connecting to your instance, please see our [connection documentation](#).

Close

12. 화면의 정보를 이용하여 인스턴스에 원격접속합니다. (Public IP 를 할당하지 않으셨다면 Public DNS 가 제공되지 않습니다. 이 경우 Elastic IP 를 생성하여 인스턴스에 할당한 뒤 EIP 를 통해 접속합니다.)

13. Windows 사용자는 원격데스크톱 (실행 -> mstsc 입력)을 사용하고 Mac 사용자는 App Store에서 Microsoft Remote Desktop 을 설치하여 접속합니다.

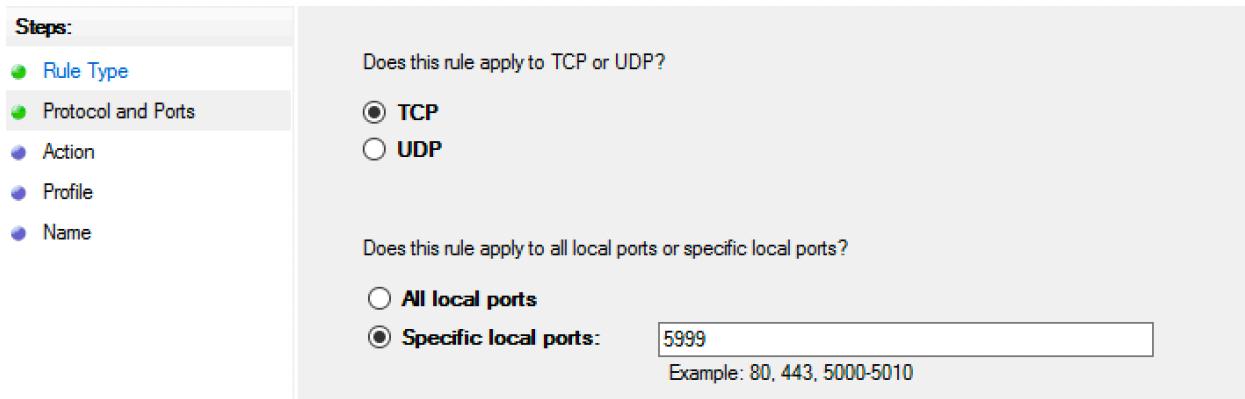
14. 방화벽 설정을 해줍니다. Control Panel -> System and Security -> Windows Firewall 메뉴로 이동합니다.



15. 왼쪽 메뉴의 Advanced settings 로 이동합니다.

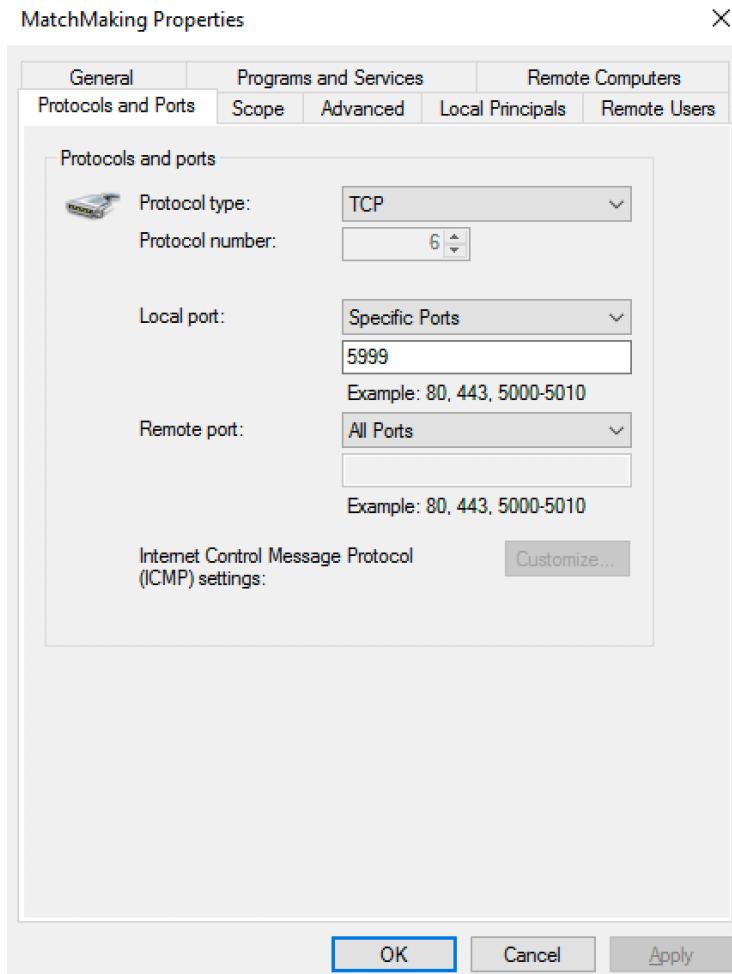
16. 왼쪽 메뉴의 Inbound Rules 를 선택하고 오른쪽의 New Rule... 메뉴를 클릭합니다. New Inbound Rule Wizard 가 시작됩니다.

17. Rule Type 은 Port 를 선택하고 Next 를 클릭합니다. Specific local ports 에 5999 를 입력한 뒤 Next 를 클릭합니다.



18. 이후는 기본값으로 진행합니다. 마지막으로 정책 이름을 입력하신 뒤 Finish 를 클릭하여 방화벽 설정을 완료합니다.

Full Stack Game Demo
Serverless Full Stack with Gomoku



19. 이전 실습으로 돌아가서 Matchmaker 서버 구성을 완료합니다.