



Full Stack Game Demo

Serverless Full Stack with Gomoku

September 2017

Table of Contents

Overview of Game.....	4
Architecture Diagram	5
Let's start to stack the full stack!	6
Section 1: DynamoDB and ElastiCache – Redis and SQS. With a dash of IAM roles.....	6
Section 2: Lambda Lambda Lambda.....	19
Section 3: Face the users with API Gateway and S3.....	27
Section 4: Putting all together in GameLift and Matchmaking Server	34
Section 5: Let's play the client	46
Appendix A Deployment Package with Python cheat sheet.....	49
Appendix B Notes on Compiling the source binary.....	50
Appendix C AWS Cli environment notes	51
Appendix D Setting up Windows notes.....	52

HoL Material by

Version 1.0 Sungsoo Khim 2017 Sep

Version 1.1 Sungsoo Khim 2017 Oct

Version 1.1 KR Hyobin An, 2017 Oct

Version 1.2 KR Seungmo Koo, 2017 Oct

Based on material prepared by Seungmo Koo 2017 Aug

Full Stack Game Demo

Serverless Full Stack with Gomoku

Hands-on-Lab 실습용 Asset 다운로드

<https://s3.ap-northeast-2.amazonaws.com/gamelifthol/Distribution.zip>

Source Code

<https://github.com/awslabs/aws-gamelift-sample>

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(...).

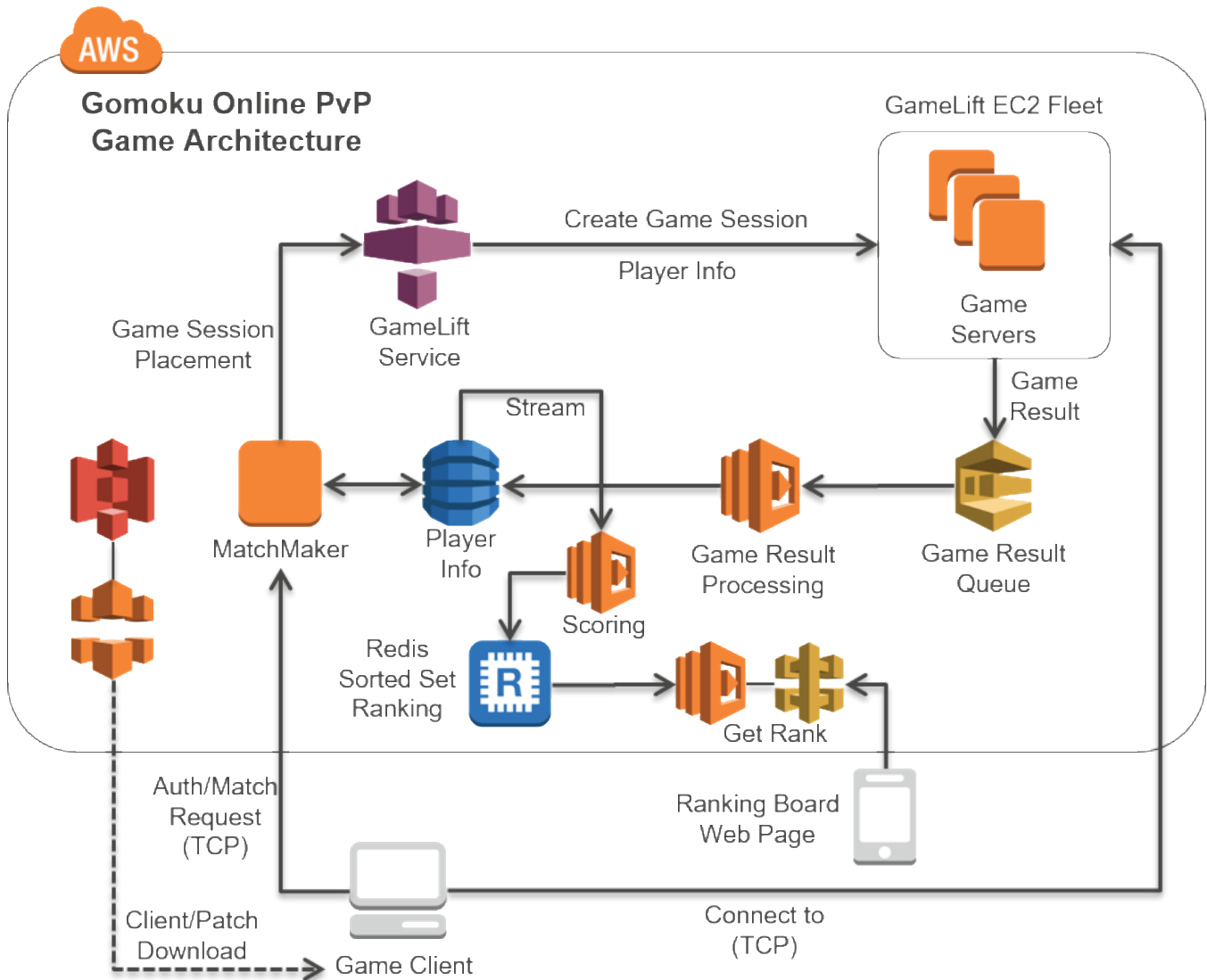
하지만 이 게임을 플레이하는 도중 사소한 오류를 발견합니다. 이 게임을 하려면 두 명의 플레이어가 있어야 하는데, 동일한 사람들과 계속 플레이를 하다 보니 게임이 재미 없다는 것입니다. 지금은 2017 년이지 1970 년대가 아닙니다. 그래서 여러분은 몇 가지 AWS 의 서비스를 이용하여 "인터넷"을 통한 플레이를 제공하기로 결정했습니다. 힘들 것 같나요? 하지만 게임은 이미 준비되어 있습니다.

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, Dynamo DB, SQS, EC2, 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 랩에서 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 Lambda를 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 하고 데이터 타입은 **String**을 선택합니다. 그리고 나서 **Create** 버튼을 클릭합니다.

Create DynamoDB table

Tutorial ?

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*

GomokuPlayerInfo

i

Primary key*

Partition key

PlayerName

String

i

☐ 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 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.

Cancel

Create

5. 테이블이 생성된 뒤에는 Stream을 활성화 합니다. 기본적으로 stream은 비활성화 되어 있습니다.

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	-

Manage Stream

View type

☐ Keys only - only the key attributes of the modified item

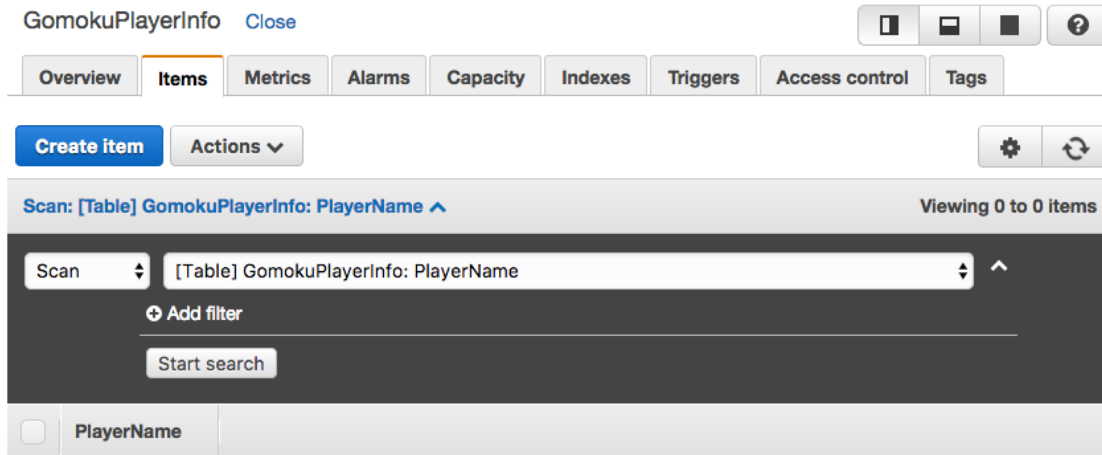
☐ New image - the entire item, as it appears after it was modified

☐ Old image - the entire item, as it appeared before it was modified

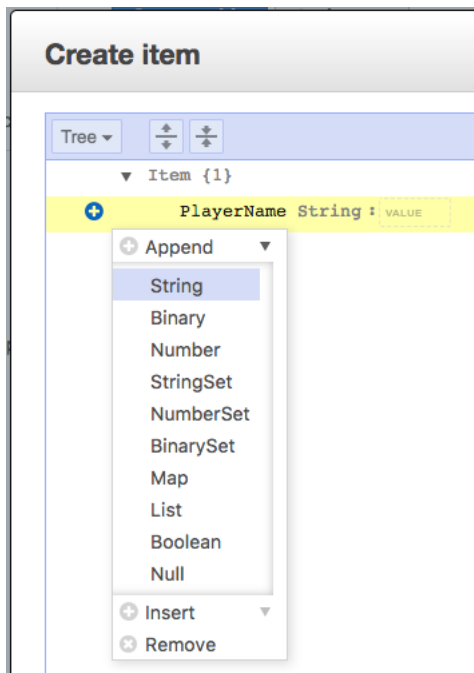
☒ New and old images - both the new and the old images of the item

[Cancel](#) [Enable](#)

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



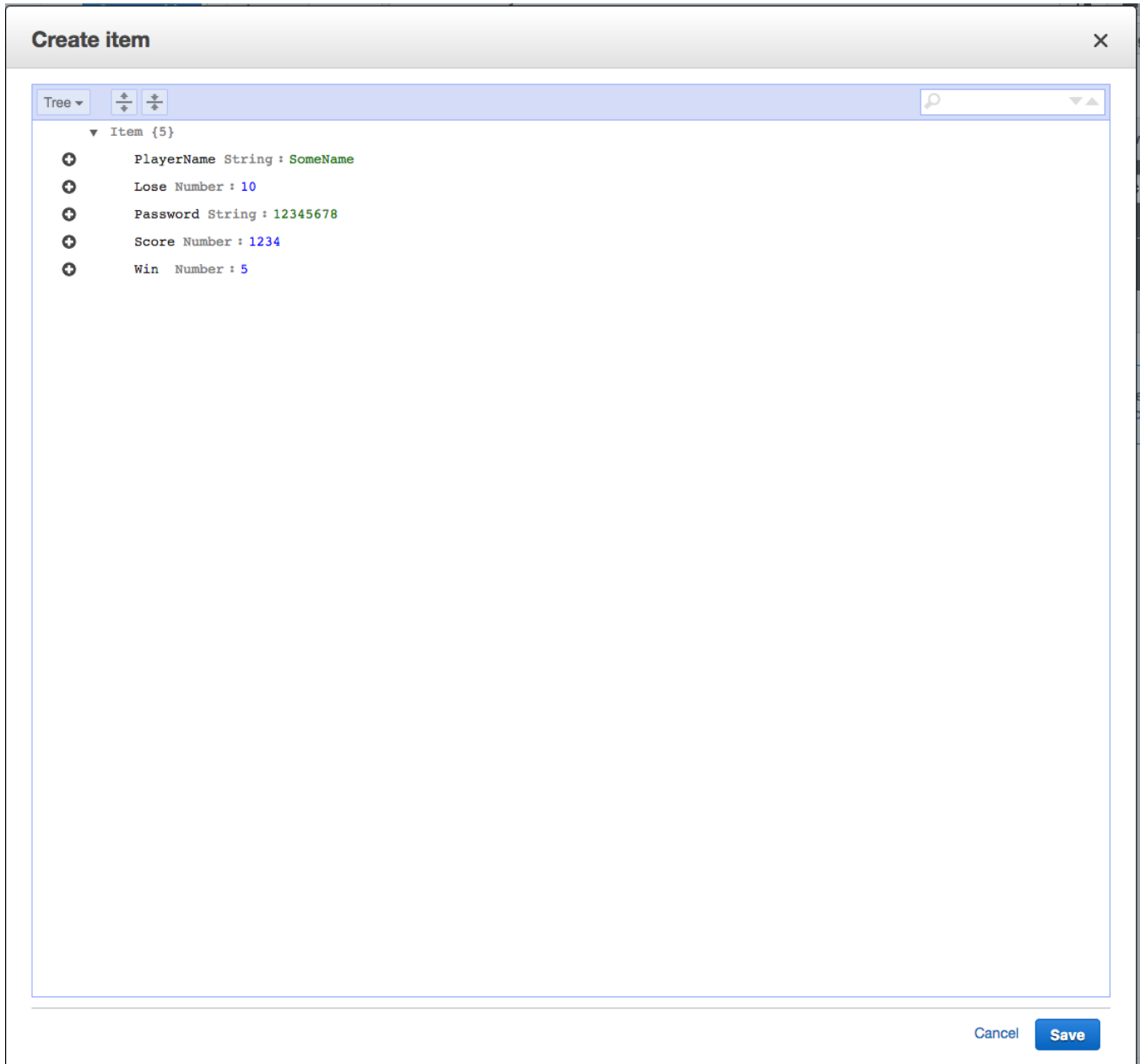
8. **Create Item** 을 클릭하여 새로운 항목을 만들어줍니다.
9. 편집기에서 + 버튼을 클릭하고, **Append**를 선택하여 추가합니다.



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

Full Stack Game Demo

Serverless Full Stack with Gomoku



(*) 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 (Scale Out)

☐ **Memcached**

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

Redis settings

Name gomokuranking ⓘ

Engine version compatibility 3.2.4 ⓘ

Port 6379 ⓘ

Parameter group default.redis3.2 ⓘ

Node type cache.t2.medium (3 GiB) ⓘ

Number of replicas None ⓘ

▶ Advanced Redis settings

Cancel

Create

Name: gomokuranking

Engine: 3.2.x (As of 2017 Aug, 3.2.4)

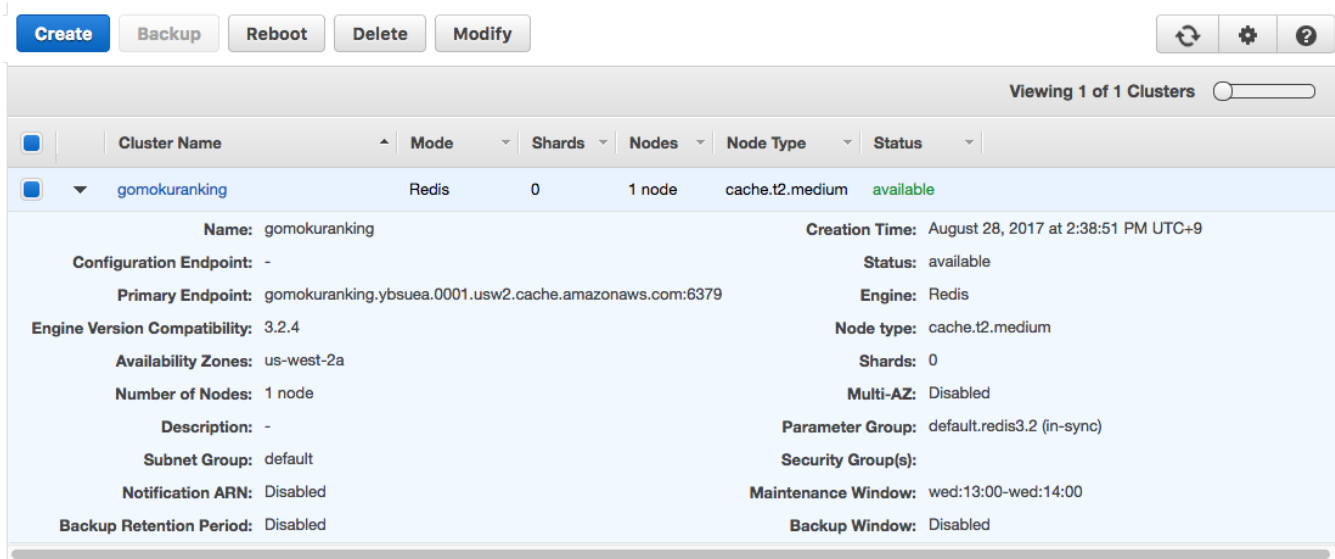
Port: 6379 (default)

Parameter group: default

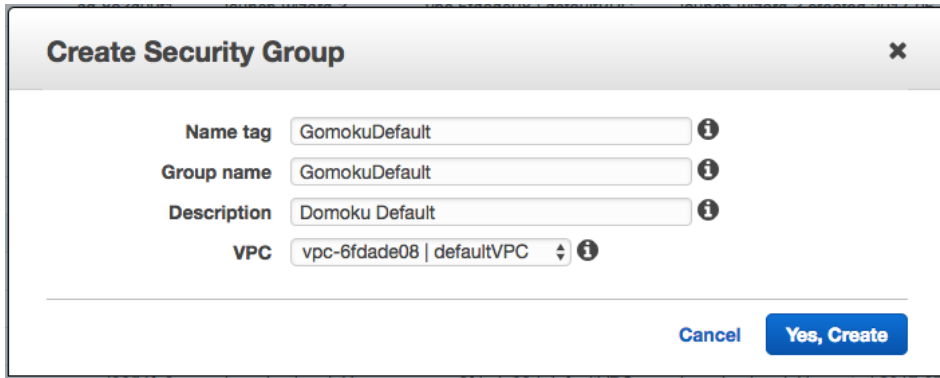
Node type: t2.medium

Number of replicas: None

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



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



Create Security Group [X]

Name tag: GomokuDefault ⓘ

Group name: GomokuDefault ⓘ

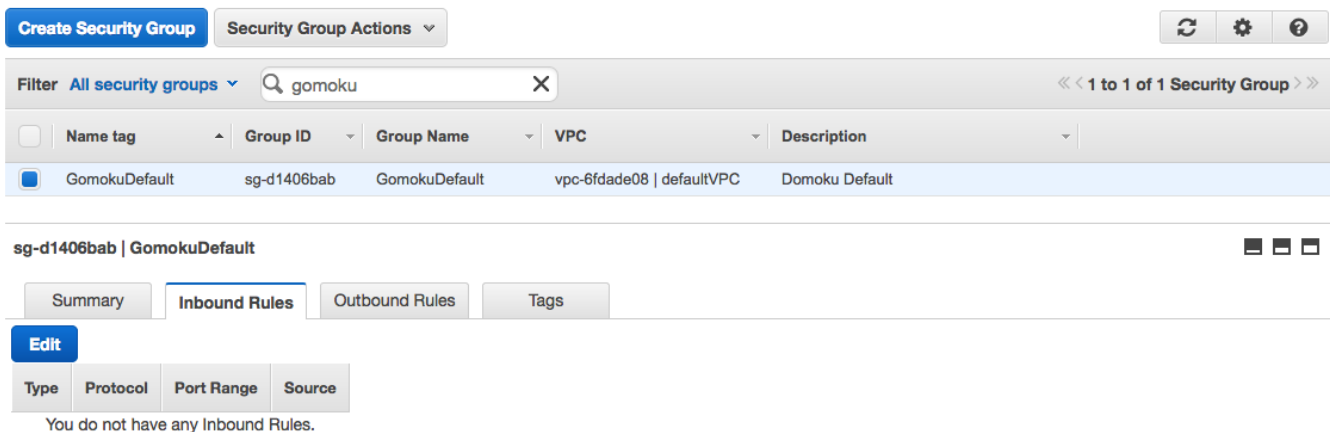
Description: Domoku Default ⓘ

VPC: vpc-6fdade08 | defaultVPC ⓘ

Cancel Yes, Create

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.



Create Security Group Security Group Actions [Refresh] [Settings] [Help]

Filter: All security groups [Search: gomoku] [X] << 1 to 1 of 1 Security Group >>

<input type="checkbox"/>	Name tag	Group ID	Group Name	VPC	Description
<input checked="" type="checkbox"/>	GomokuDefault	sg-d1406bab	GomokuDefault	vpc-6fdade08 defaultVPC	Domoku Default

sg-d1406bab | GomokuDefault [Menu] [Refresh] [Help]

Summary **Inbound Rules** Outbound Rules Tags

Edit

Type	Protocol	Port Range	Source
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.

Full Stack Game Demo

Serverless Full Stack with Gomoku

The screenshot shows the AWS IAM console interface for configuring a security group rule. At the top, there's a 'Create Security Group' button and a 'Security Group Actions' dropdown. Below this is a filter bar with 'All security groups' and a search box containing 'gomoku'. A table lists security groups, with 'GomokuDefault' (ID: sg-d1406bab) selected. Below the table, the 'sg-d1406bab | GomokuDefault' page is shown with tabs for 'Summary', 'Inbound Rules', 'Outbound Rules', and 'Tags'. The 'Inbound Rules' tab is active, displaying a table with one rule. The rule has the following configuration:

Type	Protocol	Port Range	Source	Remove
ALL Traffic	ALL	ALL	sg-d1406bab	

Below the table is an 'Add another rule' button. At the top of the rule configuration area, there are 'Cancel' and 'Save' buttons.

Type: All traffic

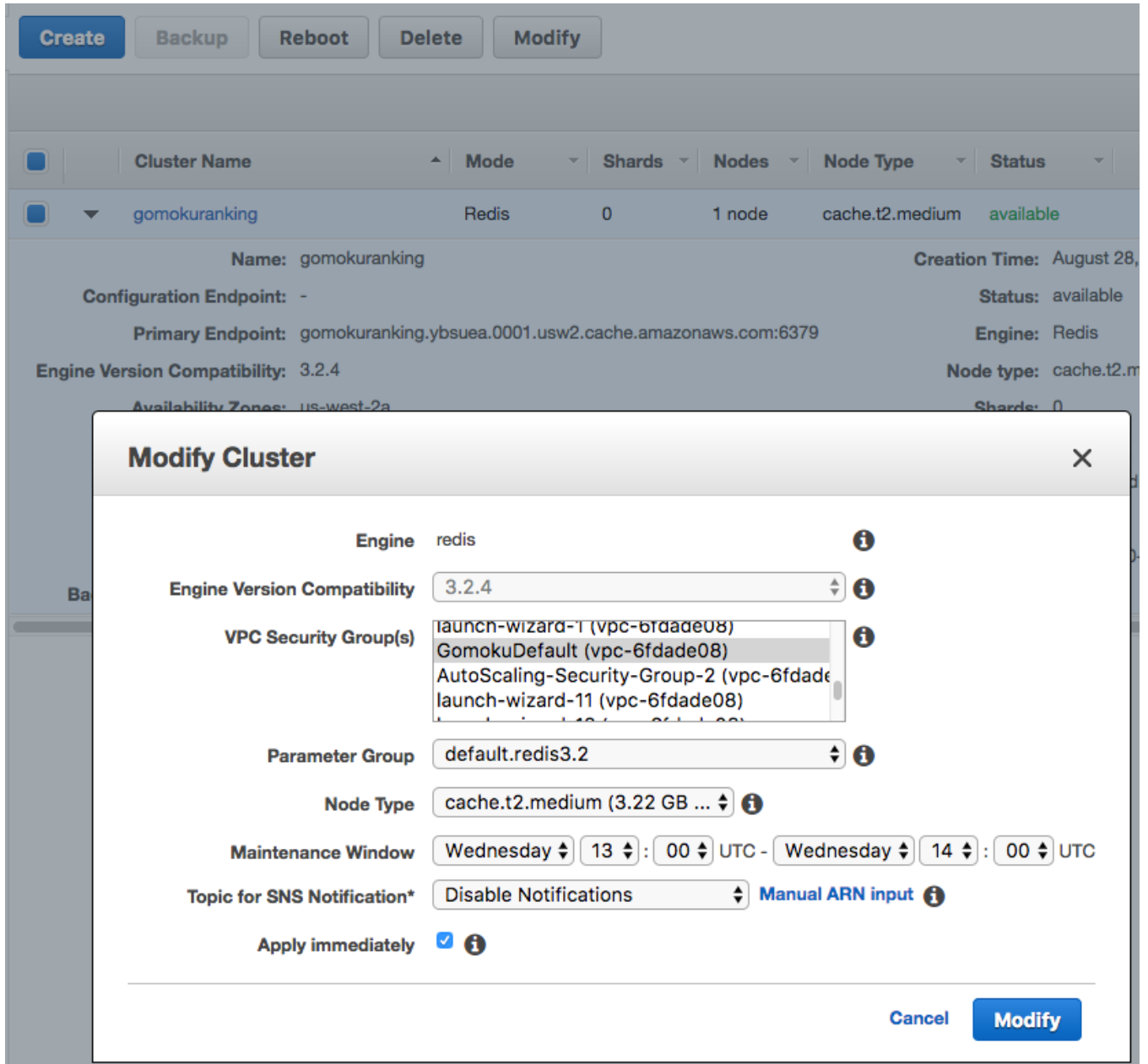
Protocol: All

Source: Security Group itself

13. 보안 그룹 생성을 완료하였다면 다시 ElastiCache 페이지로 돌아와 생성한 Redis 클러스터를 선택합니다.
14. 클러스터를 선택하고 상단의 **Modify** 버튼을 클릭합니다.

Full Stack Game Demo

Serverless Full Stack with Gomoku



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. Queue 이름은 **game-result-queue**로 입력하고, Standard Queue를 선택한 뒤 **Configure Queue** 버튼을 클릭합니다. (해당 리전에 SQS를 처음 만들게 되면 바로 SQS 생성 화면으로 넘어갑니다)
3. Queue 설정은 default를 사용합니다. **Create Queue** 버튼을 클릭하여 Queue 생성을 완료합니다.
4. Queue 생성이 완료되면 Details에 보이는 endpoint URL을 기록해둡니다. 마찬가지로 뒤의 Lambda 소스 코드에 사용할 예정입니다.

The screenshot shows the AWS Management Console interface for the 'game-result-queue' SQS queue. At the top, there's a table with columns: Name, Queue Type, Content-Based Deduplication, Messages Available, Messages in Flight, and Created. The row for 'game-result-queue' shows it is a Standard queue with N/A for deduplication, 0 messages available, 0 in flight, and created on 2017-08-28 14:40:33 GMT+09:00.

Below the table, the 'Details' tab is selected, showing the following information:

- 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.

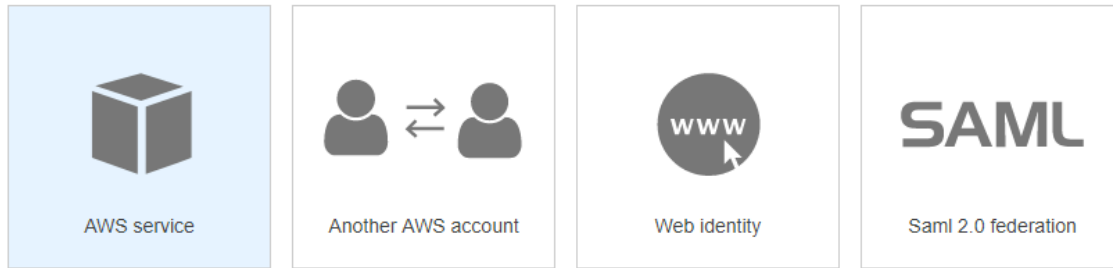
마지막으로, 이번 실습에서 만드는 full stack application에 사용할 IAM role을 생성합니다.

1. 우선, 추 후 Lambda 함수에 사용할 3 가지 역할(Role)을 생성할 것입니다. 콘솔에서 **IAM** 메뉴로 이동합니다. <https://console.aws.amazon.com/iam>
2. IAM 콘솔에서 **Role** 메뉴로 이동한 뒤 **Create role** 버튼을 클릭합니다.
3. 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.

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

Create role

Step 1 : Select role type

Step 2 : Establish trust

Step 3 : Attach policy

Step 4 : Set role name and review

Attach Policy

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type ▾		SQS			Showing 2 results
	Policy Name ▾	Attached Entities ▾	Creation Time ▾	Edited Time ▾	
<input checked="" type="checkbox"/>	 AmazonSQSFullAccess	0	2015-02-07 03:41 UTC+0900	2015-02-07 03:41 UTC+0900	
<input type="checkbox"/>	 AmazonSQSReadOnlyAccess	0	2015-02-07 03:41 UTC+0900	2015-02-07 03:41 UTC+0900	

5. 그 다음에는 **AmazonDynamoDBFullAccess** 정책 및 **AWSLambdaBasicExecutionRole** 정책을 선택한 뒤 **Next Step** 버튼을 클릭합니다.

Full Stack Game Demo

Serverless Full Stack with Gomoku

Create role

Step 1 : Select role type






Step 2 : Establish trust

Step 3 : Attach policy

Step 4 : Set role name and review

Attach Policy

Select one or more policies to attach. Each role can have up to 10 policies attached.

Filter: Policy Type ▾ Dynamo		Showing 6 results		
	Policy Name ↕	Attached Entities ↕	Creation Time ↕	Edited Time ↕
<input type="checkbox"/>	DynamoDBAutoscalePolicy	1	2017-06-20 14:40 UTC+0900	2017-06-20 14:40 UTC+0900
<input checked="" type="checkbox"/>	 AmazonDynamoDBFullAccess	0	2015-02-07 03:40 UTC+0900	2017-06-29 08:23 UTC+0900
<input type="checkbox"/>	 AmazonDynamoDBFullAccess...	0	2015-02-07 03:40 UTC+0900	2015-11-12 11:17 UTC+0900
<input type="checkbox"/>	 AmazonDynamoDBReadOnly...	0	2015-02-07 03:40 UTC+0900	2017-06-13 06:11 UTC+0900
<input type="checkbox"/>	 AWSLambdaDynamoDBExecu...	0	2015-04-10 00:09 UTC+0900	2015-04-10 00:09 UTC+0900
<input type="checkbox"/>	 AWSLambdaInvocation-Dyna...	0	2015-02-07 03:40 UTC+0900	2015-02-07 03:40 UTC+0900

6. Role name은 **Gomok-game-sqs-process**를 입력합니다. (AWS Console의 최신 UI 업데이트 상황에 따라 일부 웹 인터페이스 변경되었을 수 있습니다.)

Create role

Step 1 : Select role type

Step 2 : Establish trust

Step 3 : Attach policy

Step 4 : Set role name and review

Set role name and review

Review the following role information. To edit the role, click an edit link, or click **Create role** to finish.

Role name	<input type="text" value="Gomok-game-sqs-process"/>
	Maximum 64 characters. Use alphanumeric and '+,=,@-_' characters
Role description	<input type="text" value="Allows Lambda Function to call AWS services on your behalf."/>
	Maximum 1000 characters.
Trusted entities	The identity provider(s) <code>lambda.amazonaws.com</code>
Policies	arn:aws:iam::aws:policy/AmazonSQSFullAccess arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess
	Change policies

7. **Create role** 버튼을 클릭하여 첫 번째 역할 생성을 완료합니다.
8. 두 번째 역할도 첫 번째와 동일한 방법으로 생성합니다. 하지만 이번에는 **AmazonDynamoDBFullAccess, AmazonVPCFullAccess, AWSLambdaBasicExecutionRole** 정책을 추가합니다.
9. 두 번째 역할의 이름은 **Gomok-game-rank-update**로 지정합니다.
10. 마지막 세 번째 Role도 동일한 방법으로 생성합니다. 이번에는 **VPC Full Access, AWSLambdaBasicExecutionRole** 정책을 추가해줍니다.
11. 마지막 역할의 이름은 **Gomok-game-rank-reader**로 지정합니다.
12. 모두 생성이 완료되었다면 다음 스크린 캡처와 같이 확인할 수 있습니다.

Create new role		Role actions ▾	↺ ⚙ ?	
Gomok			Showing 3 results	
<input type="checkbox"/>	Role name ↕	Description	Creation Time ↕	
<input type="checkbox"/>	Gomok-game-rank-reader	Allows Lambda Function to call AWS services on ...	2017-08-28 18:27 UTC+0900	
<input type="checkbox"/>	Gomok-game-rank-update	Allows Lambda Function to call AWS services on ...	2017-08-28 18:25 UTC+0900	
<input type="checkbox"/>	Gomok-game-sqs-process	Allows Lambda Function to call AWS services on ...	2017-08-28 18:12 UTC+0900	

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 랩에서는 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. **Role** 은 Choose an existing role을 선택하고 기존에 만들어둔 **Gomok-game-sqs-process**를 선택하고 Create function을 실행합니다.
6. 생성이 완료되면 Triggers 탭으로 이동합니다. Add trigger를 누릅니다. CloudWatch Events를 Lambda의 실행 트리거로 선택합니다. 빈 박스를 클릭한 뒤 **CloudWatch Events**를 선택한 뒤 **Create a new rule** 버튼을 선택합니다.



Full Stack Game Demo

Serverless Full Stack with Gomoku

Configure triggers

You can choose to add a trigger that will invoke your function.

Add trigger Remove

Rule
Pick an existing rule, or create a new one.

Create a new rule ▼

Select or create a new rule

Rule name*
Enter a name to uniquely identify your rule.

Rule description
Provide an optional description for your rule.

Rule type
Trigger your target based on an event pattern, or based on an automated schedule.

☐ Event pattern

☒ Schedule expression

Schedule expression*
Self-trigger your target on an automated schedule using Cron or rate expressions. Cron expressions are in UTC.

e.g. rate(1 day), cron(0 17 ? * MON-FRI *)

Lambda will add the necessary permissions for Amazon CloudWatch Events to invoke your Lambda function from this trigger. [Learn more](#) about the Lambda permissions model.

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

☐

Cancel Previous Next

7. Rule name은 선호하는 이름을 입력해주고 Rule typed은 **Schedule expression**을 선택합니다. 우리는 이 Lambda 함수가 1분 마다 수행되길 원합니다. 따라서, Schedule expression에는 **rate(1 minute)**을 입력해줍니다.
8. 마지막으로 **Enable trigger**에 체크한 뒤 **Submit**버튼을 클릭합니다.

9. 다시 Configuration 탭으로 돌아와 아래의 생성할 함수의 정보를 참고하여 Lambda 함수를 작성합니다. (함께 배포된 파일에 있는 GameResultProcessing.py에서 표시된 변수를 적합하게 수정해서 넣어주는 과정입니다)

Name: game-sqs-process

Runtime: Python 2.7

Code: GameResultProcessing.py 파일의 내용을 Copy&Paste합니다. 코드 내의 **queue_url**은 이전 섹션에서 생성하였던 SQS URL로 변경합니다. **region_name** 부분은 여러분이 랩을 수행하는 리전으로 변경합니다. (예: ap-northeast-1)

Role: Choose an Existing Role

Role name: Gomok-game-sqs-process (이전섹션에서 생성한 것)

Advanced settings: 128MB Memory and 1 min timeout

queue_url : 위에서 생성된 SQS의 endpoint URL입니다.

region_name : 여러분이 사용하시는 리전의 코드입니다.

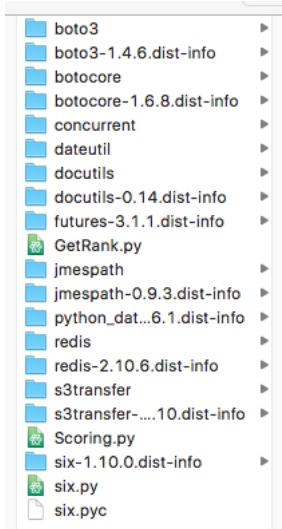
Handler: **index.lambda_handler**

10. **Save** 버튼을 클릭하여 함수를 생성합니다. 생성한 함수는 SQS 에 기록된 게임 결과 점수를 읽어와 DynamoDB 에 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. 제공되는 Zip 파일을 열어보시면 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.py파일은 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 Role은 **Gomok-game-rank-update**을 선택하고 Create function을 누릅니다.
3. Configuration 탭에서 Edit code inline대신 **Upload a .ZIP file**을 선택하고 **LambdaDeploy.zip**을 업로드합니다.
4. Handler 항목에서는 **Scoring.handler** 를 입력합니다.
5. Trigger 탭에서는 Add trigger를 누르고 **DynamoDB**을 선택합니다.
6. **GomokuPlayerInfo** DynamoDB 테이블을 Trigger 테이블로 사용할 것입니다. 다른

부분은 기본값을 유지하고 Starting position은 **Trim horizon**을 선택합니다. **Enable trigger** 를 체크한 뒤 **Submit** 버튼을 클릭합니다.

Add triggerRemove

DynamoDB▶Lambda

DynamoDB table
Please select a DynamoDB table. The Lambda function will be invoked whenever this table is updated.

GomokuPlayerInfo▼

Batch size
The largest number of records that AWS Lambda will retrieve from your table at the time of invoking your function. Your function receives an event with all the retrieved records.

100

Starting position
The position in the stream where AWS Lambda should start reading. For more information, see [ShardIteratorType](#) in the Amazon Kinesis 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).

☒

CancelPreviousNext

7. 현재까지의 상황을 Configuration탭을 통하여 확인하면 다음과 같습니다.

Full Stack Game Demo

Serverless Full Stack with Gomoku

Configuration

Triggers

Monitoring

▼ Function code

Code entry type

Upload a .ZIP file ▼

Runtime

Python 2.7 ▼

Handler

Info

Scoring.handler

Function package*

Upload

LambdaDeploy.zip (6.0 MB)

For files larger than 10 MB, consider uploading via S3.

► Environment variables

► Tags

▼ Execution role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. [Learn more](#) about Lambda execution roles.

Choose an existing role ▼

Existing role

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

Gomoku-game-rank-update ▼

▼ Basic settings

Memory (MB)

Info

Your function is allocated CPU proportional to the memory configured.

192 MB

Timeout

Info

1 min 0 sec

Description

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

Copyright 2017, Amazon Web Services, All Rights Reserved

Page 24

▼ Network

VPC [Info](#)

Select a VPC that your function will access.

Default vpc-bad53ad3 (172.31.0.0/16) ▼

Subnets*

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

subnet-83af40ea (172.31.0.0/20) | ap-northeast-2a ✕

subnet-43c1e809 (172.31.16.0/20) | ap-northeast-2c ✕

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-61eb190a (GomokuDefault) | GomokuDefault ✕

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

<div> <div><</div> <div>1</div> <div>></div> </div>		
Security group ID	Ports	Source
sg-61eb190a	All	sg-61eb190a

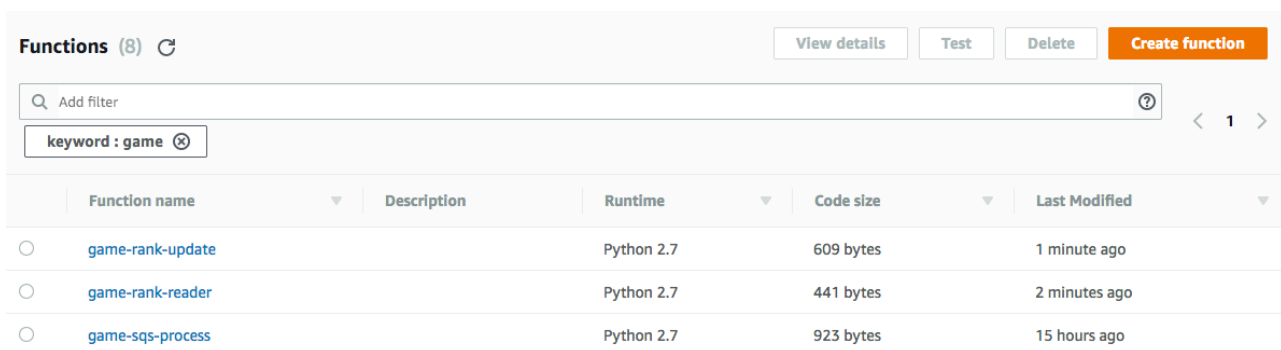
9. 위의 과정이 완료되면 Save를 눌러 함수 생성을 완료합니다. (이 단계에서 함수 Test를

누르게 되면 아직 실패합니다.)

마지막으로 세 번째 Lambda 함수를 생성하겠습니다.

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

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



	Function name	Description	Runtime	Code size	Last Modified
<input type="radio"/>	game-rank-update		Python 2.7	609 bytes	1 minute ago
<input type="radio"/>	game-rank-reader		Python 2.7	441 bytes	2 minutes ago
<input type="radio"/>	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 주기적으로 위에서 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.)

```
{ "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.)

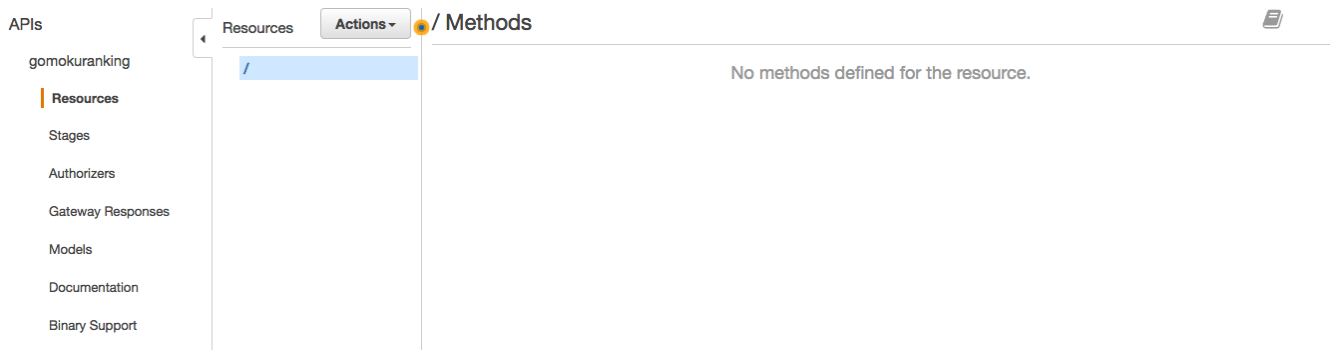
(데이터가 갱신되는 것은 위에서 만들어준 Lambda함수의 실행 주기가 1분이기 때문에 약 1분에서 2분정도까지 시간이 걸릴 수 있습니다.)

Section 3: Face the users with API Gateway and S3

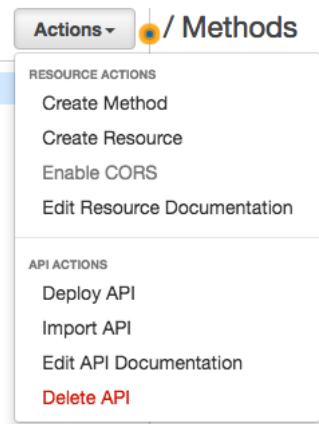
우리의 Lambda 함수 중 한가지가 API Gateway를 필요로 한다는 것을 알고 있습니다. 이것은

개발 중의 애플리케이션의 Lambda 함수를 실행할 일종의 "Gateway" 역할을 하기 때문에 API Gateway라 이름이 붙여졌습니다.

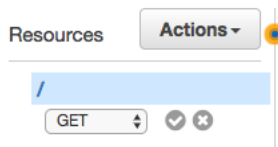
1. 우선 콘솔에서 API Gateway 메뉴로 이동합니다.
<https://console.aws.amazon.com/apigateway>
2. **New API**를 선택하고, API name은 **gomokuranking**을 입력한 뒤 **Create API** 버튼을 클릭합니다.
3. API가 생성된 뒤에는 다음과 같은 빈 화면이 보일 것입니다.



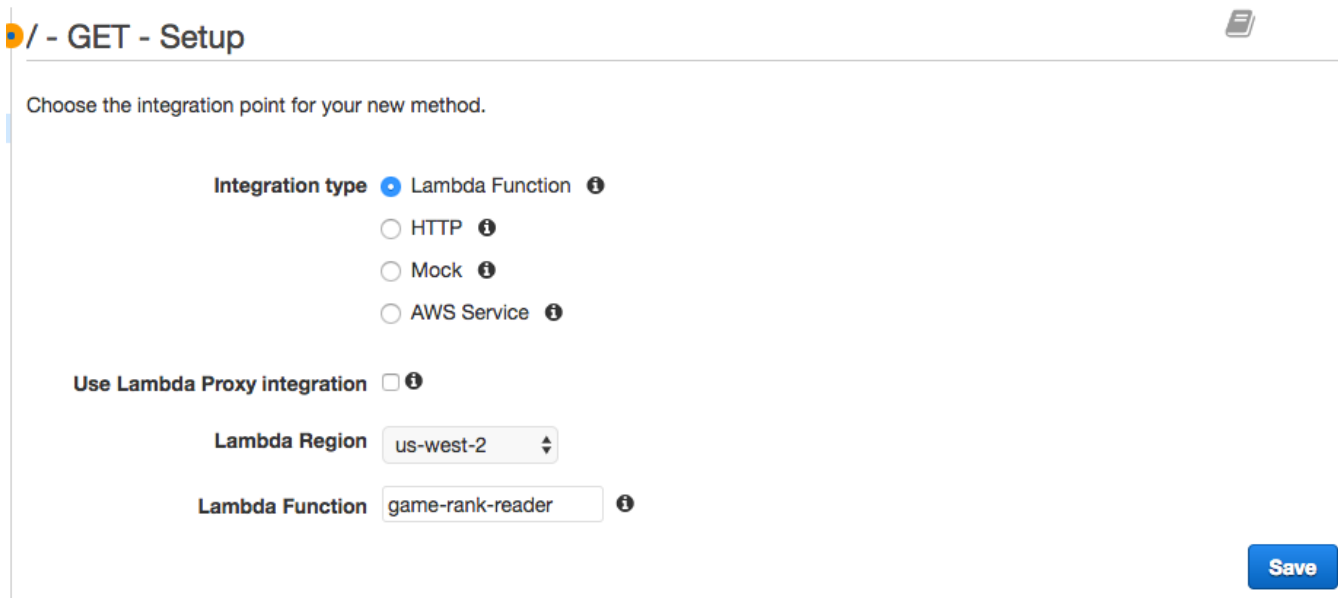
4. 처음할 작업은 새로운 Method를 생성하는 것입니다. **Actions** 버튼을 클릭한 뒤 **Create Method** 메뉴를 선택합니다.



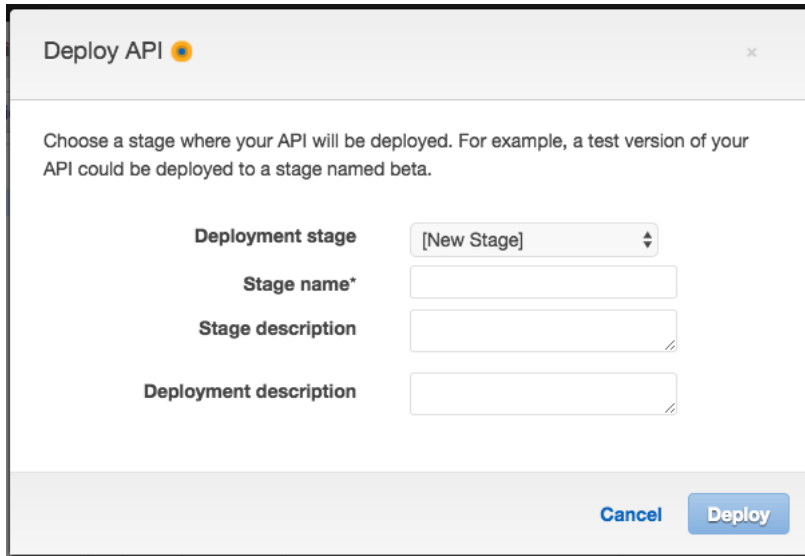
5. 아래 작은 리스트박스가 보일 것입니다. **GET**을 선택한 뒤 옆의 체크 버튼을 클릭합니다.



6. 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.



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



Deploy API

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.

Deployment stage: [New Stage]

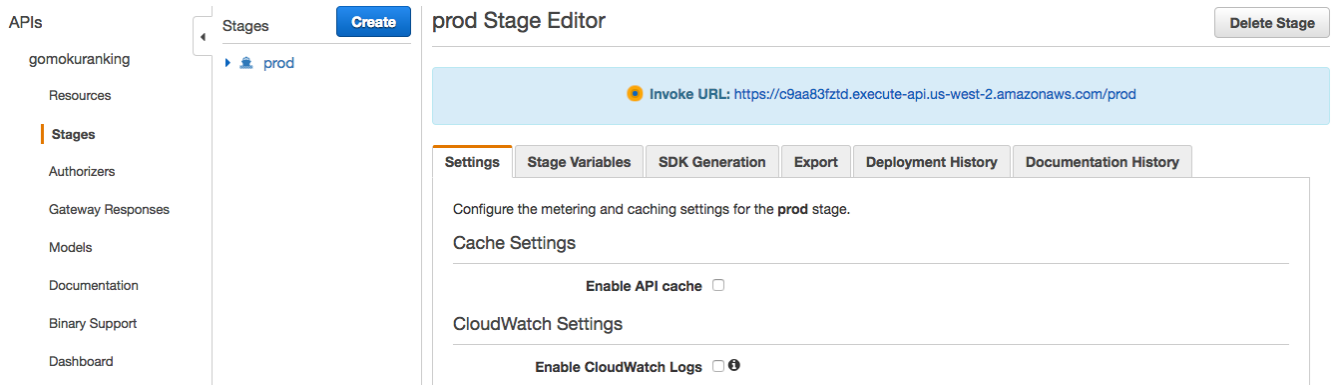
Stage name*:

Stage description:

Deployment description:

[Cancel](#) [Deploy](#)

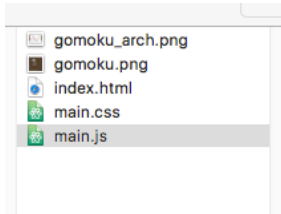
8. **[New Stage]** 를 선택하고 Stage name에는 **prod**를 입력합니다. **Deploy** 버튼을 클릭하여 진행합니다.
9. 완료되면 다음 스크린 캡처와 같이 Stage 구성이 된 것을 확인할 수 있습니다. Invoke URL을 기록해둡니다. 이 후 S3를 이용한 정적 웹페이지 구성에 사용됩니다.



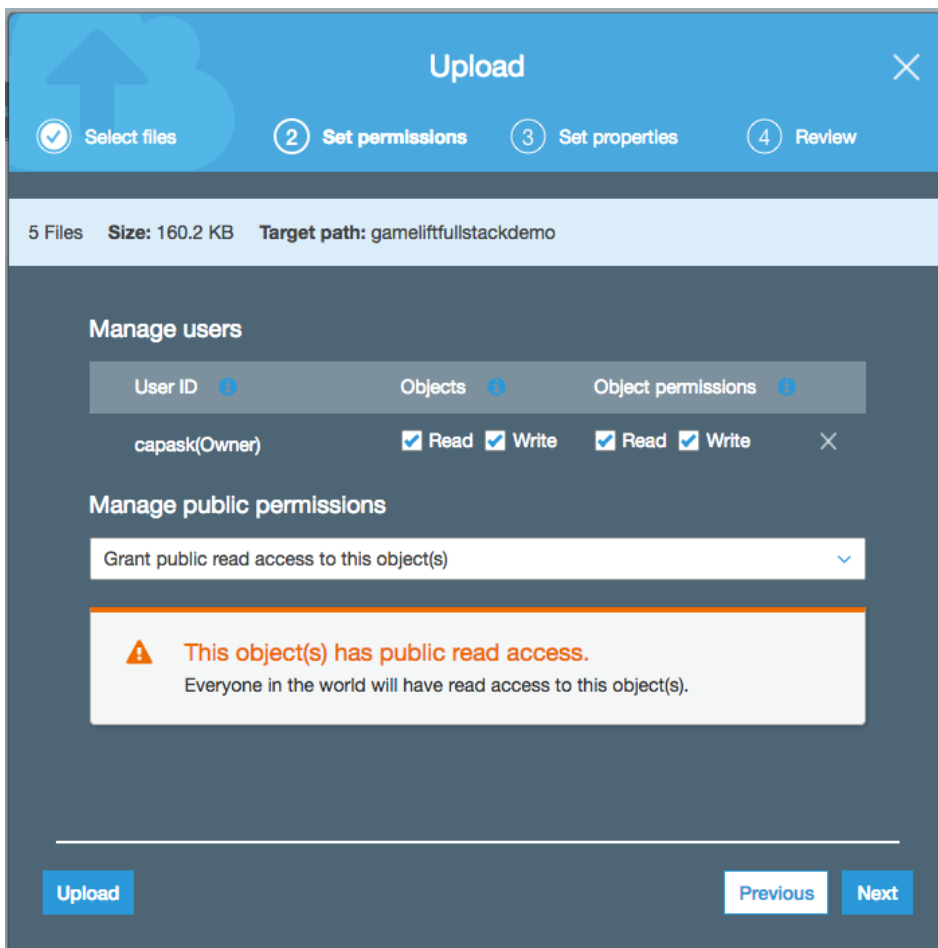
이제 S3가 호스팅하는 웹 사이트를 생성할 것입니다. 버킷에 Ranking board html파일과 Javascript 파일을 업로드하는 것만으로 쉽게 웹 사이트를 호스팅할 수 있습니다.

1. S3 서비스로 이동하여 웹 사이트에 사용할 S3 버킷을 생성합니다.
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.

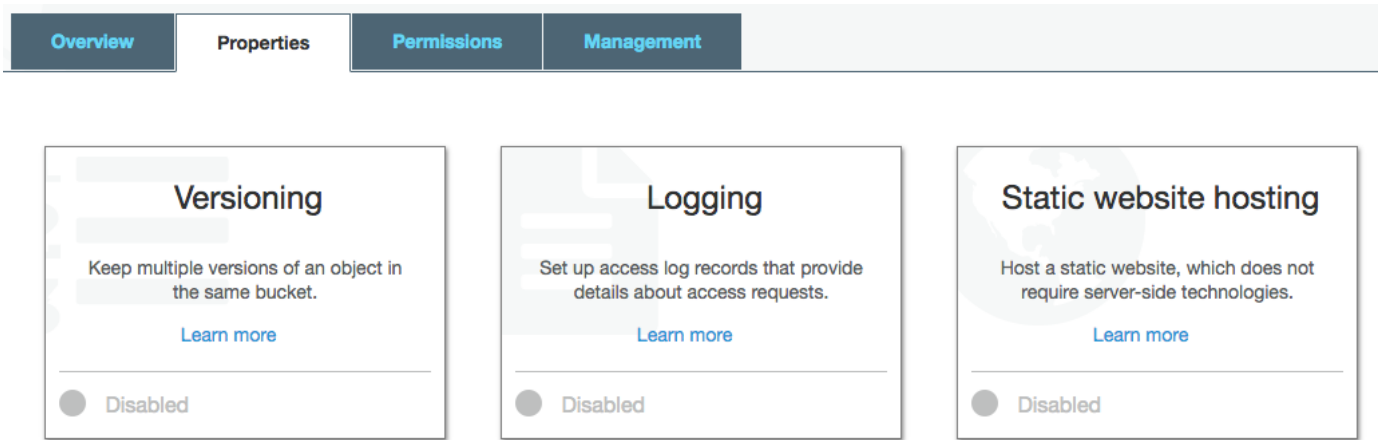
- 수정한 main.js를 저장한 뒤 web 디렉토리의 파일을 전부 앞서 생성한 버킷에 업로드해줍니다.



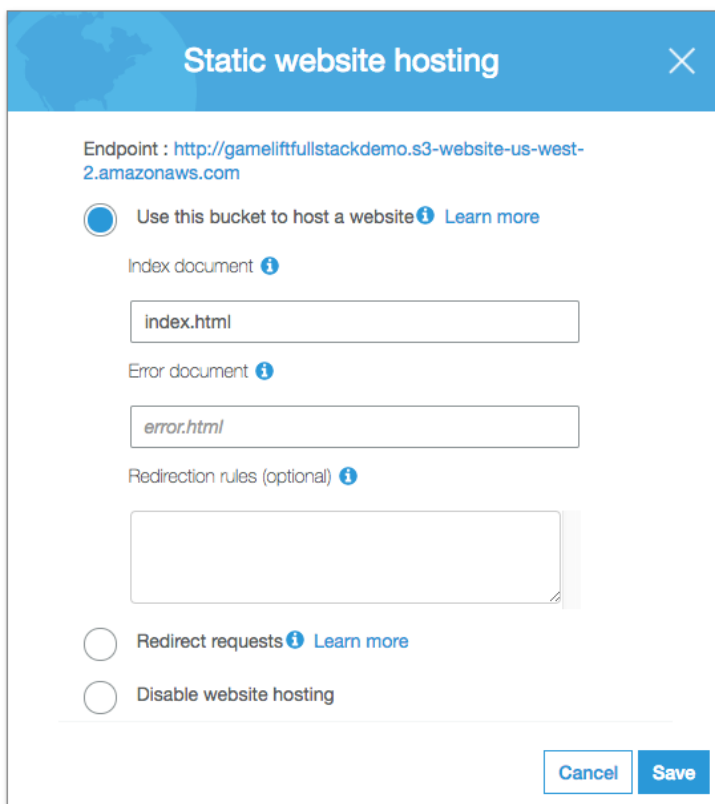
- 업로드할 때 Public read access 권한을 부여합니다.



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



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

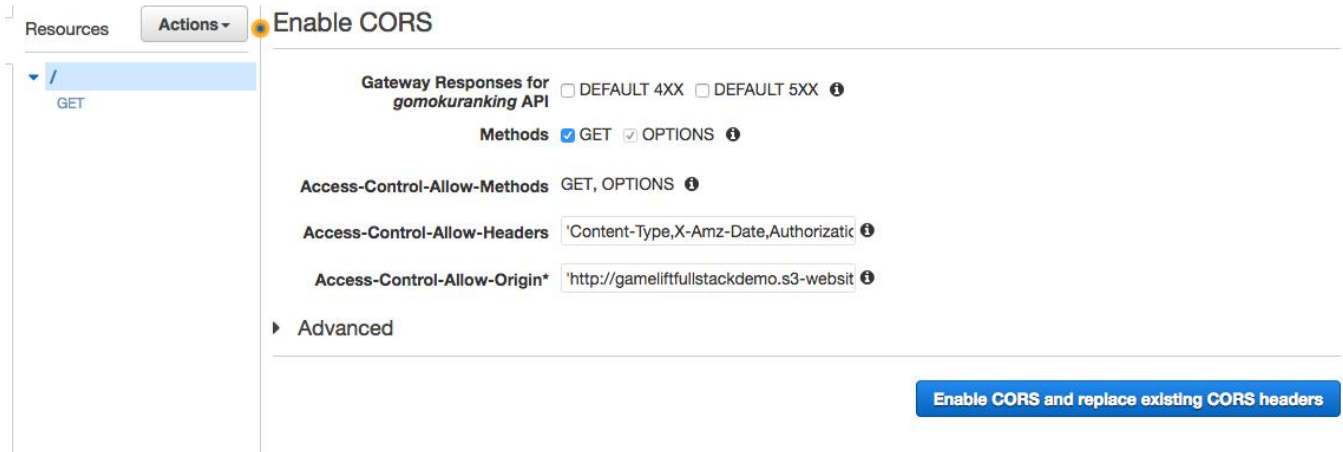


8. 파일 업로드를 완료하면 Static website hosting의 Endpoint로 접속하여 단순한 웹 페이지를 확인할 수 있습니다.
9. API Gateway로 돌아와서 CORS 설정을 해주어야 합니다. 이를 통해 Ranking board

정보를 웹 페이지에서 읽어 올 수 있게 됩니다.

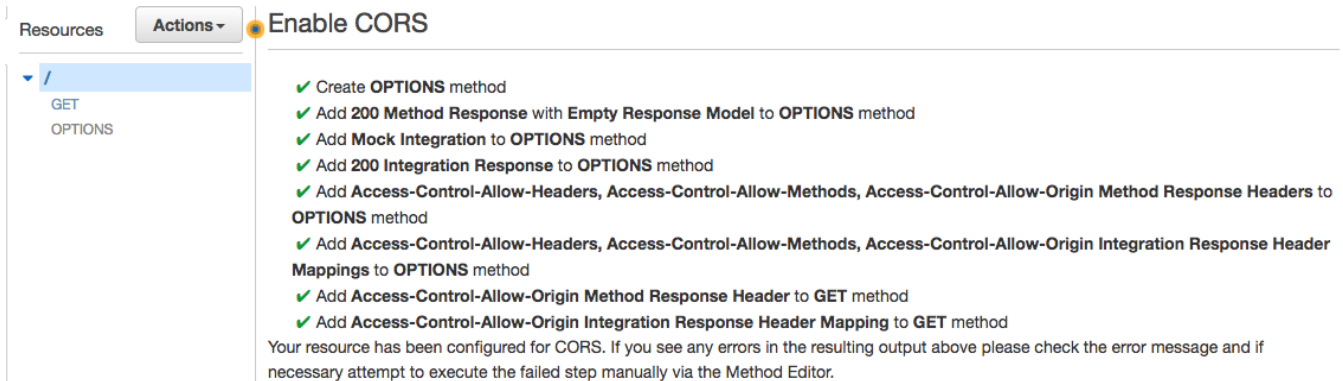
10. API Gateway로 돌아와서 **Actions** 버튼을 클릭하고 **Enable CORS** 옵션을 선택합니다.

11. CORS 페이지에서 Access-Control-Allow-Origin 에 static website URL로 수정해줍니다.
(URL 뒤의 / 가 영향을 미칠 수 있기 때문에 확실하지 않으면 기본 값인 * 로 진행합니다.)



12. **Enable CORS** 버튼을 클릭하여 진행합니다.

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



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 and Matchmaking Server

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. 우선 IAM user를 생성하여 게임 서버 바이너리가 AWS 리소스에 접근할 권한을 주어야 합니다. 생성한 IAM user는 섹션 1에서 만들었던 SQS에 접근 권한을 가져야 합니다.
2. 콘솔에서 IAM 서비스로 이동합니다. <https://console.aws.amazon.com/iam>
3. 새로운 **User**를 생성합니다. Access type은 **Programmatic access**를 선택합니다.

Add user



Set user details

You can add multiple users at once with the same access type and permissions. [Learn more](#)

User name* GomokServer

+ Add another user

Select AWS access type

Select how these users will access AWS. Access keys and autogenerated passwords are provided in the last step. [Learn more](#)

- Access type* ☒ **Programmatic access**
 Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- ☐ **AWS Management Console access**
 Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

Cancel

Next: Permissions

4. **Next: Permissions** 버튼을 클릭합니다.
5. Permissions 페이지에서 **Attach existing policies directly** 옵션을 선택합니다. 그리고 **SQSFullAccess** 권한을 체크합니다.

Full Stack Game Demo

Serverless Full Stack with Gomoku

Add user



Set permissions for GomokServer

Add user to group

Copy permissions from existing user

Attach existing policies directly

Attach one or more existing policies directly to the user or create a new policy. [Learn more](#)

Create policy Refresh

Filter: Policy type Showing 2 results

	Policy name	Type	Attachments	Description
<input type="checkbox"/>	AmazonSQSFullAcc...	AWS managed	1	Provides full access to Amazon SQS via the A...
<input type="checkbox"/>	AmazonSQSReadO...	AWS managed	0	Provides read only access to Amazon SQS via ...

Cancel Previous Next: Review

6. 진행하여 User 생성을 완료합니다.
7. User가 생성되면 해당 User의 **Access key ID**와 **Secret access key**를 확인할 수 있습니다. 두 가지 모두를 기록해둡니다.

	User	Access key ID	Secret access key
▶	✓ GomokServer	AKIAIPFBY6SUDXXC4TCQ	***** Show

Close

8. CSV 파일도 다운로드합니다.
9. 이제 바이너리 파일들을 준비합니다. 이번 실습에는 이미 컴파일된 파일이 준비되어 있지만, 직접 컴파일 하고 싶으실 경우 Appendix B를 참조할 수 있습니다.
10. GomokuServer.exe, aws-cpp-sdk-*.dll, config.ini, install.bat, aws-cpp-sdk-gamelift-server.dll 파일들이 GomokServer 폴더에서 확인할 수 있습니다.
11. Distribution 패키지에 함께 제공되는 **vc_redist.x64.exe** 파일을 반드시 위의 **GomokServer**폴더에 복사해 주세요
12. 텍스트 편집기를 통하여 config.ini 파일을 수정합니다. 파일에는 총 4군데 구성 요소가 있습니다. SQS_REGION는 SQS를 생성한 Region입니다. (예: ap-northeast-1)
SQS_ACCESSKEY 와 SQS_SECRETKEY 에는 앞서 생성한 IAM User의 Key 정보를 입력합니다. 마지막으로 SQS_ENDPOINT 에는 SQS의 Endpoint를 입력합니다. 파일을 저장한 뒤 편집기를 닫습니다 (아래의 스크릿샷 처럼 따옴표 없이 입력해주세요.)

[config]

```
# GameResult SQS
SQS_REGION = ap-northeast-2
SQS_ACCESSKEY = AKIAJYQSSKCTVAH4UAAA
SQS_SECRETKEY = eFh4eBcaMvlfVvkZ48acudfdfuWz2cX3Boi03eQmuh
SQS_ENDPOINT = https://sqs.ap-northeast-2.amazonaws.com/675967543641/game-result-queue
```

13. Gamelift는업로드의 복잡성 때문에 현재 CLI를 통한 업로드 만을 지원합니다. AWS CLI 환경이 구성되어 있지 않다면 Appendix C를 참고하여 구성합니다.
14. 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-1
15. 업로드할 때 실습 Region을 올바르게 설정하였는지 확인해야 합니다.
16. 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.

Create fleet from build

Delete build



Filter: Status: All

Viewing 1 build(s)

	Status	Name	Build ID	Version	OS	Si...	Date cre...	Fi...
<input type="radio"/>	Ready	GomokuServer-Build-1	build-06b01eca-5411-4...	1.0.0	Windows 2012	14.95 MB	2017-08-29 16:33:...	0

17. 콘솔의 빌드 페이지에서 방금 업로드한 빌드를 선택합니다. Build 버튼의 **Create fleet** 버튼을 클릭합니다. 이를 통해 게임 서버의 fleet을 생성하게 됩니다.

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

Name: GomokuGameServerFleet-1

Instance Type: C3.large

Full Stack Game Demo

Serverless Full Stack with Gomoku

Create fleet



Fleet details

Create a new fleet from any build that you have uploaded to Amazon GameLift.

Name*	GomokuGameServerFleet-1	?
Description		?
Metric group		?
Build*	GomokuServer-Build-1 1.0.0 2017-08-29 16:33:58 UTC+0900	?
Operating system	Windows 2012 R2	
Build ID	build-06b01eca-5411-4cad-a378-a24683a6f79c	
Build size	14.95 MB	

Instance type

Select an instance type to determine the computing resources that will be used to run your dedicated game servers on this fleet. All Amazon GameLift instances include 50 GB of elastic block storage (EBS). Instances for this fleet are either Windows or Linux, depending on the operating system of the selected build. For more details, see [EC2 Instance Types](#). Instance type cannot be changed once the fleet is created.

	Family	Instance type	vCPU	Memory (GB)	Instance sto...	Network per...
<input checked="" type="radio"/>	Compute optimized	c3.large Free tier	2	3.75	50GB EBS only	Moderate
<input type="radio"/>	General purpose	t2.micro	1	1	50GB EBS only	Low to moderate
<input type="radio"/>	General purpose	t2.small	1	2	50GB EBS only	Low to moderate
<input type="radio"/>	General purpose	t2.medium	2	4	50GB EBS only	Low to moderate
<input type="radio"/>	General purpose	t2.large	2	8	50GB EBS only	Low to moderate
<input type="radio"/>	General purpose	m4.large	2	8	50GB EBS only	Moderate
<input type="radio"/>	General purpose	m4.xlarge	4	16	50GB EBS only	High
<input type="radio"/>	General purpose	m4.2xlarge	8	32	50GB EBS only	High
<input type="radio"/>	General purpose	m4.4xlarge	16	64	50GB EBS only	High
<input type="radio"/>	General purpose	m4.10xlarge	40	160	50GB EBS only	10 Gigabit

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

Concurrent processes: 50

우측의 녹색 체크 버튼을 클릭하여 확인합니다.

Max Concurrent game session activation: 50

Full Stack Game Demo

Serverless Full Stack with Gomoku

Process management - New

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		1

[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 50 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

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

Amazon GameLift

Fleets

GomokuGameServerFleet-1

Search

Q

Fleet: GomokuGameServerFleet-1

A fleet manages one build of your game server, which is replicated across many Amazon EC2 instances. Use this page to monitor your fleet.

Actions

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

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

시간이 조금 소요됩니다.

21. Fleet을 생성하는 동안에 Alias를 생성하겠습니다.

22. 메뉴에서 **Create alias** 옵션을 선택합니다. 그리고 Alias name을 입력해줍니다.

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

The screenshot shows the 'Amazon GameLift Aliases' console. At the top, there's a navigation bar with 'Amazon GameLift', 'Aliases', and a 'Create alias' button. Below this is a search bar. The main section is titled 'Create alias' with a help icon. Under 'Alias details', there's a prompt to 'Provide a name and description for your alias.' There are two input fields: 'Alias name*' with the value 'GomokuAlias' and 'Description'. Below this is the 'Routing options' section, which includes a description: '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.' There are two fields: 'Type*' with a dropdown set to 'Simple' and 'Associated fleet*' with a dropdown set to 'GomokuGameServerFleet-1'. At the bottom right, there are 'Cancel' and 'Configure alias' buttons.

24. 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 U

25. 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. (인스턴스를 새로 기동하는데는 시간이 조금 소요됩니다. 따라서 이런 케이스에는 클라이언트와 MatchMaking 서버의 상황을 고려해야 합니다. 이번 실습에서는 이런 케이스를 다루지는 않습니다.)

우리는 Gamelift가 우리의 게임서버로 동작하도록 구성할 것입니다. 이제 Matchmaking 서버를 생성하여 우리의 클라이언트가 연결하여 게임을 할 수 있도록 합니다. 이번에도 이미 컴파일된 바이너리 파일을 사용하겠습니다.

1. 이전 스텝과 다르게 MatchMaker폴더 내의 config.ini 파일을 먼저 수정하도록 하겠습니다. 편집기로 열어주세요.
2. 이번에도 4개의 구성 요소를 수정해야 합니다. GAMELIST_ALIAS는 생성한 Gamelift의 Alias입니다. LISTEN_PORT는 Matchmaking 서버가 사용할 포트입니다. Region 에는 Gamelift 스택이 실행 중인 Region으로 수정합니다. PLAYER_TABLENAME에는 섹션 1에서 생성한 DynamoDB 테이블 이름으로 수정합니다. (아래의 스크린샷을

참고하세요)

```
[config]

# GameLift Endpoint Alias
GAMELIFT_ALIAS = alias-62d2bff3-dc3a-4c70-8373-80c69cb413ec

# MatchMaker Listen port
LISTEN_PORT = 5999

# GameLift Region
GAMELIFT_REGION = ap-northeast-2

# DDB TableName
PLAYER_TABLENAME = GomokuPlayerInfo
```

3. Matchmaking 서버가 계속 실행되어야 하기 때문에 EC2 Windows 인스턴스를 생성하여 서버 애플리케이션을 실행할 것입니다. EC2 인스턴스가 사용할 IAM 역할을 하나 생성합니다.
4. IAM 서비스로 이동하여 **Roles** 메뉴에서 **Create new role** 버튼을 클릭합니다.
5. Role type은 AWS Service 의 EC2를 선택합니다.
6. 정책은 **DynamoDB, Gamelift**가 필요합니다. 우선 **AmazonDynamoDBFullAccess** 를 선택한 뒤 **Next Step**을 클릭합니다.
7. Role name 에는 나중에 확인할 수 있는 이름을 준 뒤 **Create** 버튼을 클릭합니다.
8. Roles 페이지에서 생성한 **GomokuMatchMaker Role**을 선택한 뒤 **GameLift** 정책을 추가합니다.

Full Stack Game Demo

Serverless Full Stack with Gomoku

IAM > Roles > GomokuMatchMaker

▼ Summary

Role ARN	arn:aws:iam::483463947488:role/GomokuMatchMaker
Role description	Allows EC2 instances to call AWS services on your behalf. Edit
Instance Profile ARNs	arn:aws:iam::483463947488:instance-profile/GomokuMatchMaker
Path	/
Creation time	2017-08-29 17:18 UTC+0900

Permissions Trust relationships Access Advisor Revoke sessions

Managed Policies ^

The following managed policies are attached to this role. You can attach up to 10 managed policies.

[Attach Policy](#)

Policy Name	Actions
AmazonDynamoDBFullAccess	Show Policy Detach Policy Simulate Policy

Inline Policies v

9. **AmazonDynamoDBFullAccess**는 이미 추가가 되어있기 때문에 **GameLift** 정책을 추가하기 위하여 **Add Inline Policies** 섹션을 클릭한 뒤 정책을 생성합니다.

Inline Policies ^

There are no inline policies to show. To create one, [click here](#).

10. 정책을 추가할 때는 Policy Generator를 사용할 수 있습니다. 이번 기회에 한번 사용해보도록 하겠습니다.

Set Permissions

Select a policy template, generate a policy, or create a custom policy. A policy is a document that formally states one or more permissions. You can edit the policy on the following screen, or at a later time using the user, group, or role detail pages.

☒ Policy Generator

Use the policy generator to create your own set of permissions. [Select](#)

☐ Custom Policy

11. Effect는 **Allow**, AWS Service는 **Amazon GameLift**를 선택합니다. Actions는 **All Actions**를 선택하고 ARN은 기본값인 * 로 진행합니다.

Edit Permissions

The policy generator enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [Overview of Policies](#) in Using AWS Identity and Access Management.

Effect ☒ Allow ☐ Deny

AWS Service

Actions

Amazon Resource Name (ARN)

☒ CreateAlias

☒ CreateBuild

☒ CreateFleet

12. **Add Statement**를 클릭한 뒤 아래의 **Next Step** 버튼을 클릭하여 진행합니다.

13. 이렇게 생성한 정책은 다음 스크린 캡처와 같습니다.

Review Policy

Customize permissions by editing the following policy document. For more information, see [Overview of Policies](#) in the *Using IAM* guide. To test the effects of this policy, use the [Policy Simulator](#).

Policy Name

policygen-GomokuMatchMaker-201708291725

Policy Document

```

1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "Stmt1503995031000",
6       "Effect": "Allow",
7       "Action": [
8         "gamelift:*"
9       ],
10      "Resource": [
11        "*"
12      ]
13    }
14  ]
15 }
```

14. **Apply Policy** 버튼을 클릭하여 **Role** 생성을 완료합니다.

Full Stack Game Demo

Serverless Full Stack with Gomoku

▼ Summary

Role ARN	arn:aws:iam::488523074720:role/GomokuMatchMaker	
Role description	Allows EC2 instances to call AWS services on your behalf.	Edit
Instance Profile ARNs	arn:aws:iam::488523074720:instance-profile/GomokuMatchMaker	
Path	/	
Creation time	2017-08-29 17:18 UTC+0900	

[Permissions](#) [Trust relationships](#) [Access Advisor](#) [Revoke sessions](#)

Managed Policies

The following managed policies are attached to this role. You can attach up to 10 managed policies.

[Attach Policy](#)

Policy Name	Actions
AmazonDynamoDBFullAccess	Show Policy Detach Policy Simulate Policy

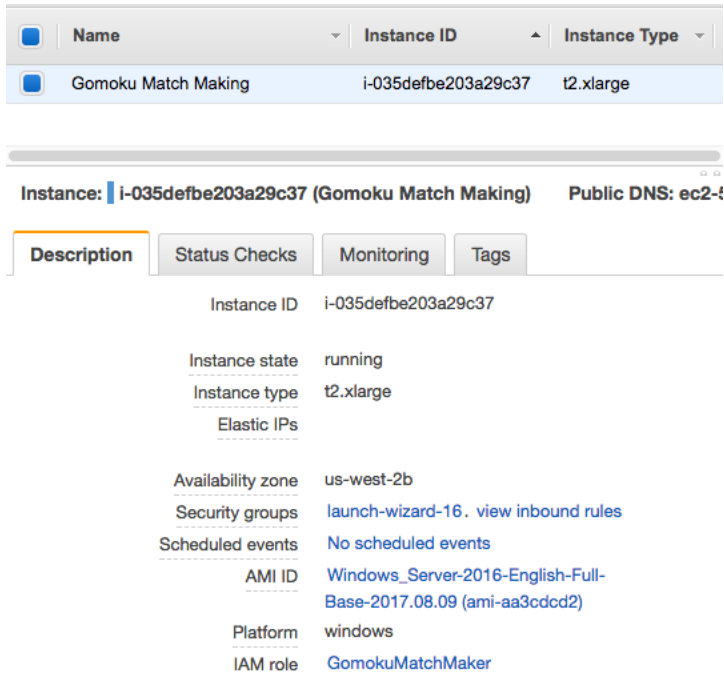
Inline Policies

This view shows all inline policies that are embedded in this role.

[Create Role Policy](#)

Policy Name	Actions
policygen-GomokuMatchMaker-201708291725	Show Policy Edit Policy Remove Policy Simulate Policy

15. EC2 서비스로 이동하여 **Microsoft Windows Server 2016 Base** 이미지를 선택하여 인스턴스를 생성합니다. 인스턴스 타입은 **t2.large**나 더 큰 타입을 선택합니다.
(Windows Server 설정이 익숙치 않다면 Appendix D를 참고하세요.)
16. Security Group은 **5999**포트를 **0.0.0.0/0** 으로 열어줍니다. 이를 통해 게임 클라이언트가 서버와 통신을 할 수 있게됩니다. **RDP 포트도 퍼블릭**으로 설정합니다.



17. RDP를 통해 생성한 인스턴스에 로그인 합니다. **Control Panel**의 **Firewall** 설정으로 이동하여 **inbound traffic**에서 **TCP 5999** 포트를 **퍼블릭**으로 열어줍니다.
18. 실습 파일의 matchmaker 폴더의 파일(dll 파일 3개, ini 파일 1개, exe 파일 2개)을 생성한 Matchmaker 서버로 복사합니다.
19. Matchmaker 서버로 복사한 파일 중 vc_redist.x64.exe 파일을 실행하여 실습에 필요한 런타임을 설치해줍니다.
20. config.ini 파일을 열어 올바른 GameLift Alias 등이 설정되었는지 다시 한번 확인합니다.
21. 여기까지 확인했다면 이제 서버를 실행하겠습니다. GomokuMatchMaker.exe 파일을 실행하여 줍니다.

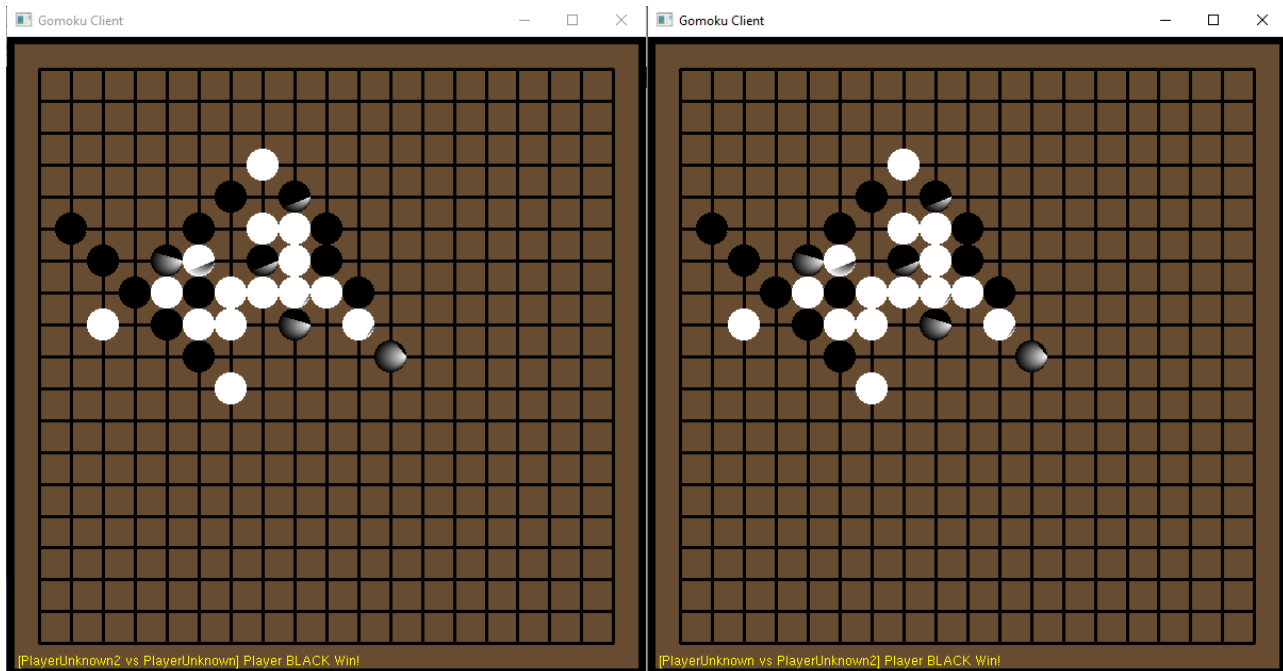
필요에 의하여 서비스의 형태로 바이너리를 실행하는 것은 여러분의 선택이 되겠습니다. Windows RDP로 접속하여 실행하였기 때문에 이후 RDP창만 닫으면 백그라운드로 계속해서 실행될것입니다.

Section 5: 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 client 폴더로 이동합니다.
2. 마찬가지로 Client폴더 내의 config.ini 파일의 수정이 필요합니다.
3. SERVER_IP는 섹션 4에서 생성한 인스턴스의 Public IP로 수정합니다. (없다면 인스턴스에 EIP를 할당해준 뒤 EIP를 사용하세요)
4. PLAYER_NAME과 PLAYER_PASSWD를 (임의로) 지정합니다.
5. 게임 클라이언트를 실행합니다. 아래의 스크립 캡처와 같이 실행될 것입니다.
6. 게임 화면에서 우클릭한 뒤 Start 하세요!
7. 다른 선수가 같은 서버에 접속할 때까지 기다려야 합니다. 그러면 Matchmaking 서버가 대전을 시작하게 됩니다. (다른 사용자가 없다면 config.ini를 수정한 복사본을 만들어 실행할 수 있습니다.)
8. 이미 졌다고 판단된다면 우클릭한 뒤 Give up을 할 수 있습니다.
9. 수고하셨습니다!



참고: MatchMaker는 기본적으로 PLAYER_NAME가 이미 DynamoDB에 존재하면 PLAYER_PASSWD를 비교합니다. 패스워드가 맞다면 로그인이 자동으로 완료 됩니다. 만일

PLAYER_NAME에 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.

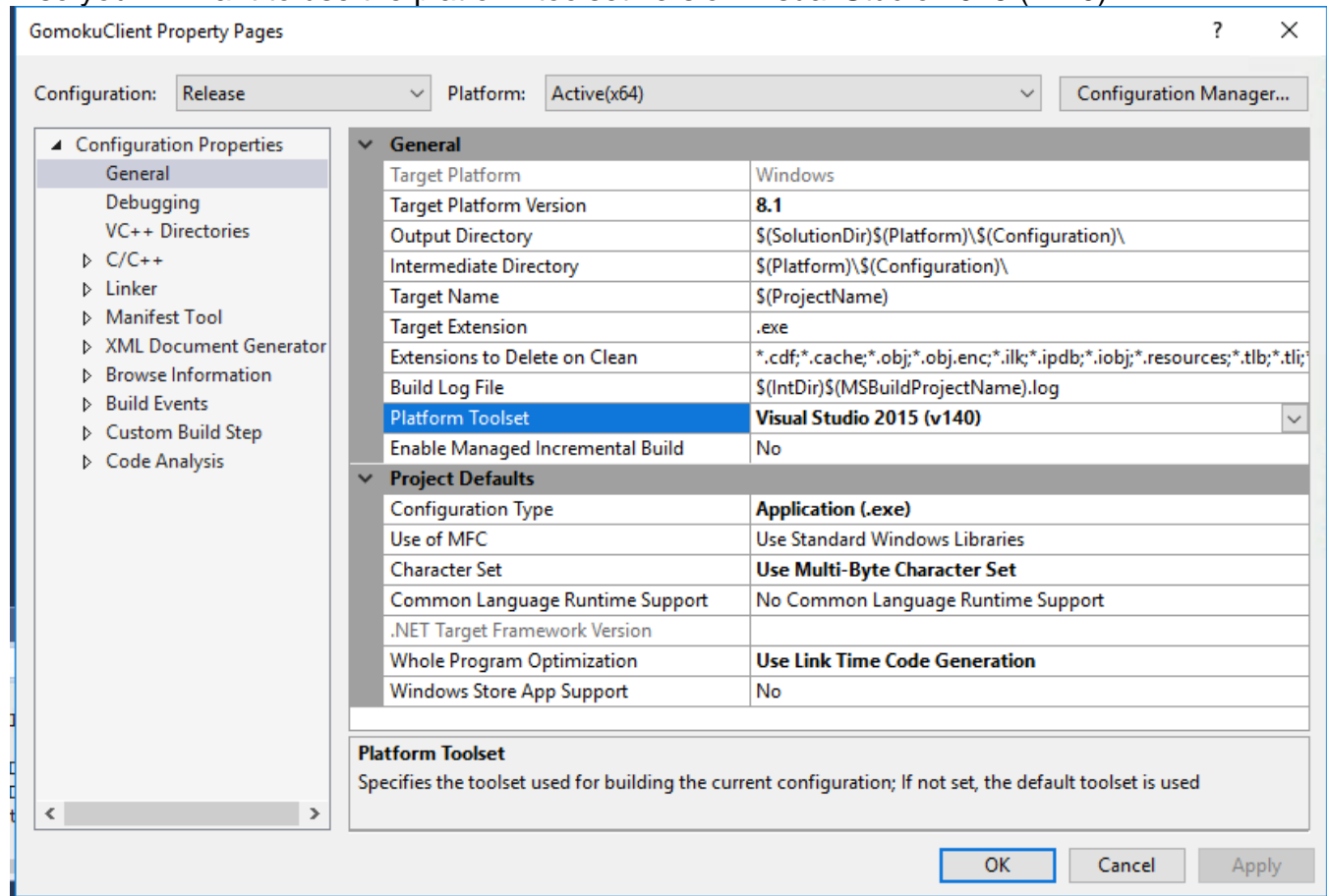
Appendix B Notes on compiling the source binary

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.

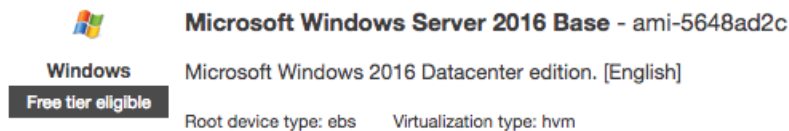
Please refer to this page for detailed instructions:

<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 and 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.

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:
 Description:

Type	Protocol	Port Range	Source
RDP	TCP	3389	Custom 0.0.0.0/0
Custom TCP	TCP	5999	Custom 0.0.0.0/0

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 사용해야 합니다.
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

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	Choose File	virginia.pem
---------------	-------------	--------------

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

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAtzITIHA0zSPuBVzA0m+dMBy4957TWpjxW5EXYFWwtf0DBuaBPGxCCWCIDwT
eqiDcXeUOYdRUNDgp8rJ/D6rjP8W8c8fefJkJtdlIaqAsVyP2y8SDkbyl8+8t5AVfQbzjGGELX
Nq6dRjtGl8q3YIDZ/8TWt5+7WkJWF4v03jjNaionbiaaE6i5Hv/mYmMPH3/Dn7qgMyodpUilJ0Qu
jKazmiZr6KwyBgokrG0WGeWfIYABfgoGc7kSJenmCYDKnkjVFjaPfrz91gPC4FNbZ9HIBK/WzJK
-----
```

Decrypt Password

Back Close

11. 다음과 같이 인스턴스 접속 정보가 화면에 나타납니다.

Connect To Your Instance

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:

Download Remote Desktop File

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

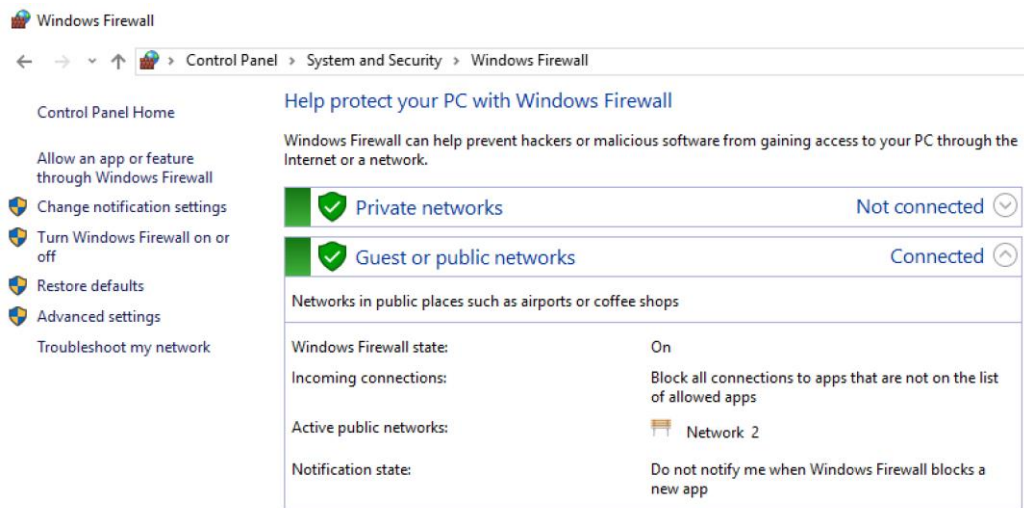
Public DNS	ec2-34-230-70-26.compute-1.amazonaws.com
User name	Administrator
Password	

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

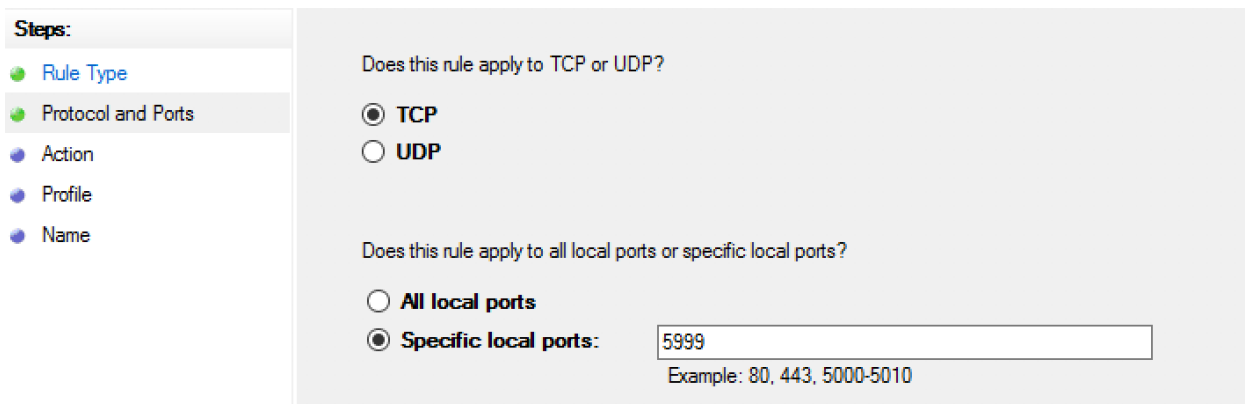
If you need any assistance connecting to your instance, please see our [connection documentation](#).

Close

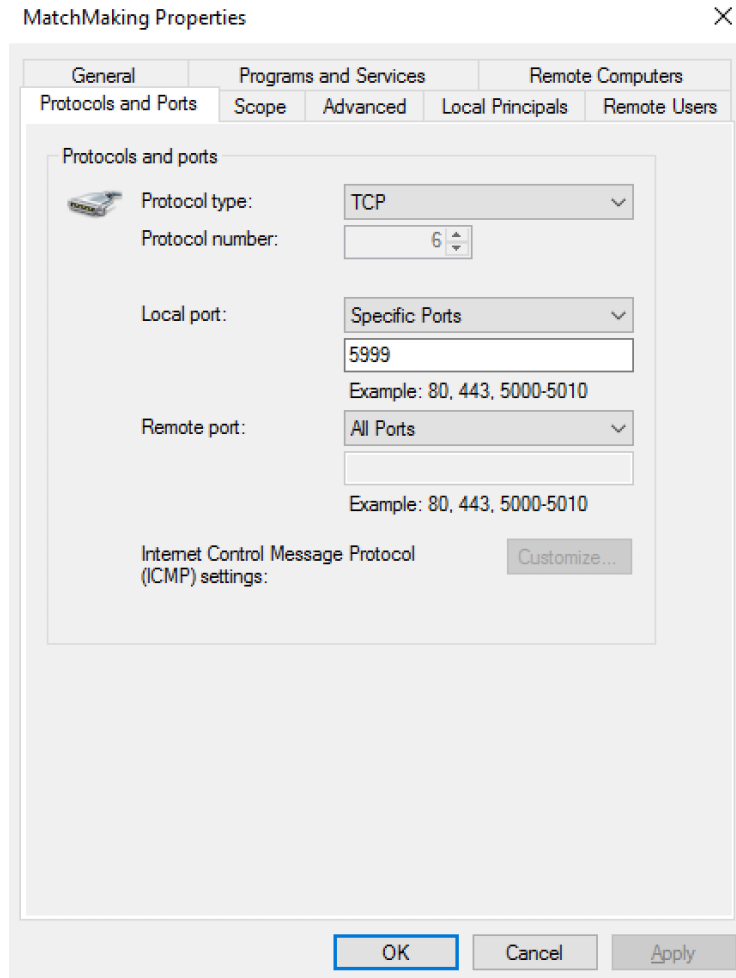
12. 화면의 정보를 이용하여 인스턴스에 원격접속합니다. (Public IP 를 할당하지 않으셨다면 Public DNS 가 제공되지 않습니다. 이 경우 Elastic IP 를 생성하여 인스턴스에 할당한 뒤 EIP 를 통해 접속합니다.)
13. Windows 사용자는 원격데스크톱 (실행 -> mstsc 입력)을 사용하고 Mac 사용자는 App Store 에서 Microsoft Remote Desktop 을 설치하여 접속합니다.
14. 방화벽 설정을 해줍니다. Control Panel -> System and Security -> Windows 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.



19. 이전 실습으로 돌아가서 Matchmaker 서버 구성을 완료합니다.