

Ask Django

AWS RDS/S3 서비스 이용하여 배포하기

일반적인 장고 배포 구성

- 웹 서버 : 장고 애플리케이션 구동
- 데이터베이스 서버
- 스토리지 서버 : static/media 파일 저장 및 서빙
- 캐시 서버 : key/value 스토어

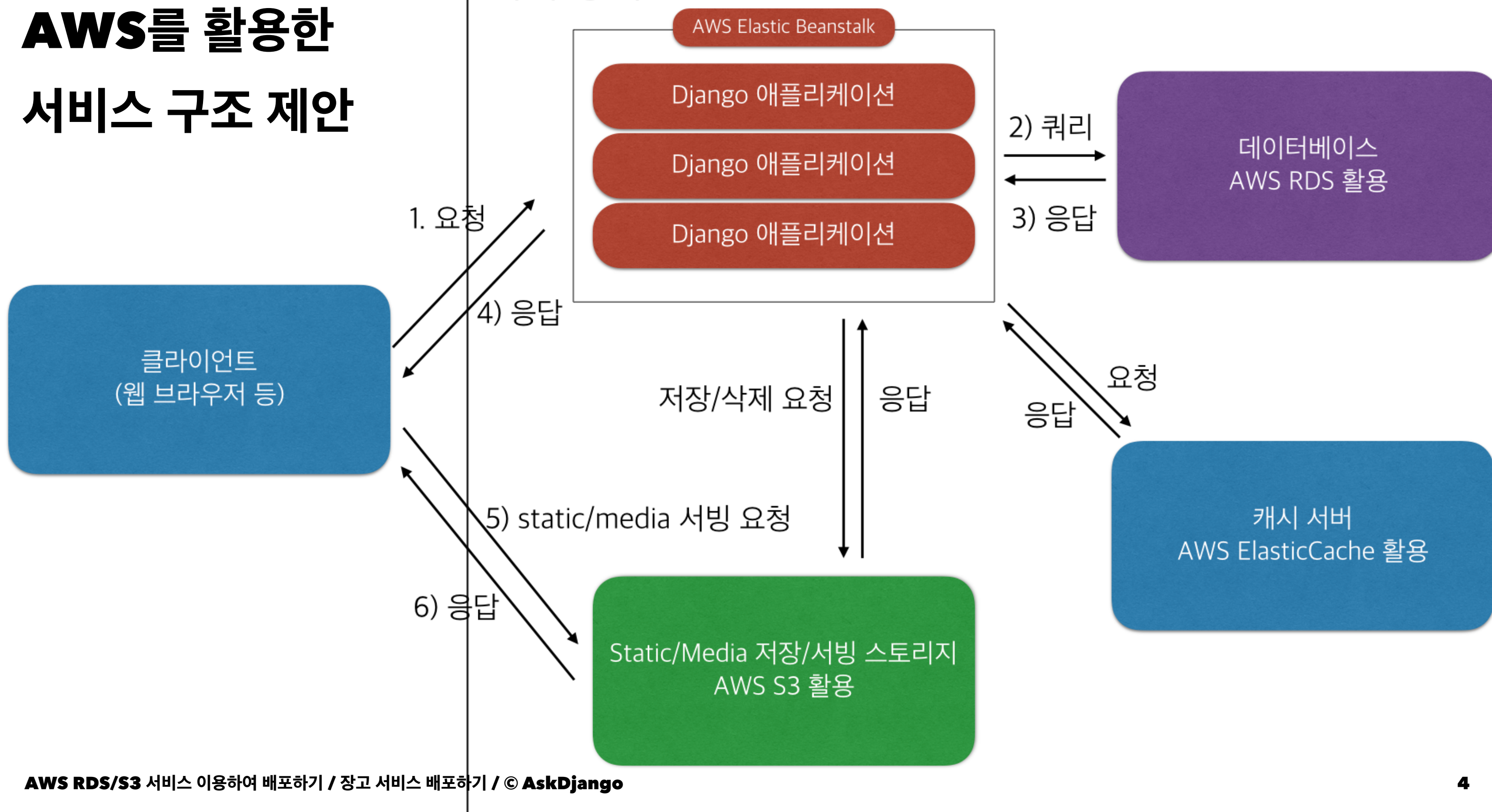
AWS 대응 선택지

관련 서비스를 통해 인프라 운영에 대한 부담을 줄여봅시다.

- 웹 서버
 - AWS EC2에 직접 설치
 - **AWS Elastic Beanstalk** : 웹 서비스 간편 배포/확장 서비스 - 서비스
 - AWS Lambda : Serverless
- 데이터베이스 서버
 - AWS EC2에 직접 설치
 - **AWS RDS** (Relational Database Service) : **Database as a Service** - 서비스
- 스토리지 서버
 - AWS EC2에 직접 저장/서빙
 - **AWS S3** (Simple Storage Service) - 서비스
- 캐시 서버
 - AWS EC2에 직접 설치
 - Elastic Cache : Redis, Memcached - 서비스

AWS를 활용한 서비스 구조 제안

서버 영역



시작하기에 앞서 프로젝트 모델에
Media 파일 필드를 만들어줍시다.

AWS RDS

Relational Database Service

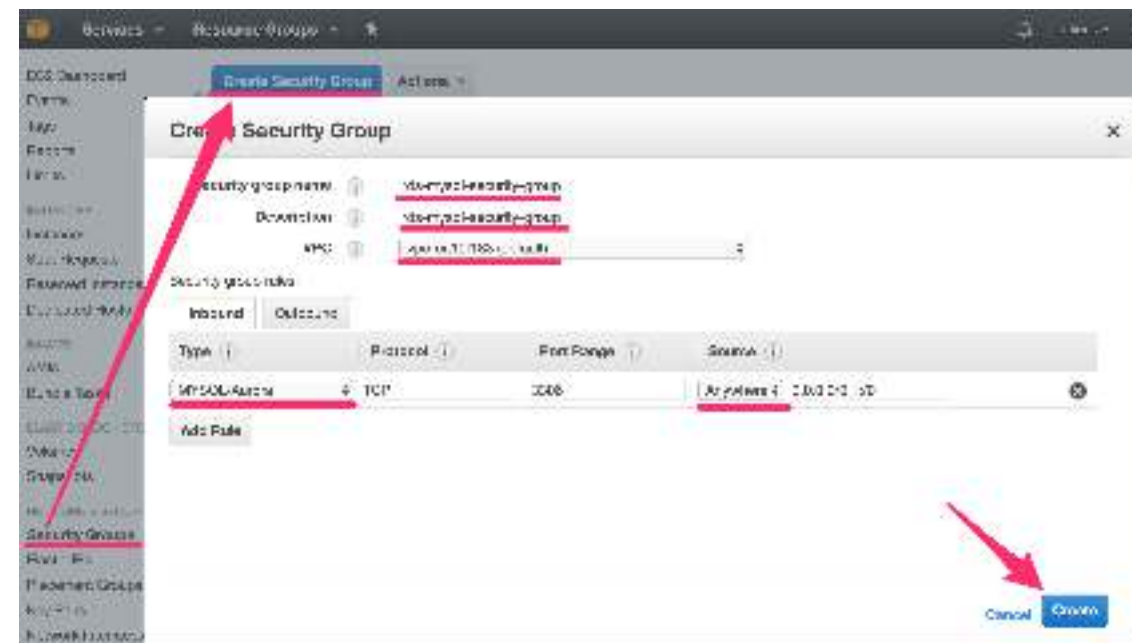
RDS #doc

- 데이터베이스 서비스
- 지원 엔진 : Amazon Aurora, **MySQL**, **MariaDB**, Oracle, Microsoft SQL Server, **PostgreSQL**
- Free tier #ref
 - 단일 AZ, db.t2.micro, 750시간/월
 - 20GB DB스토리지, 1,000만 I/O

Security Group ¹ 생성

MySQL 포트 3306 개방

심플한 설명을 위해 **Source**를 **Anywhere** 설정 ²

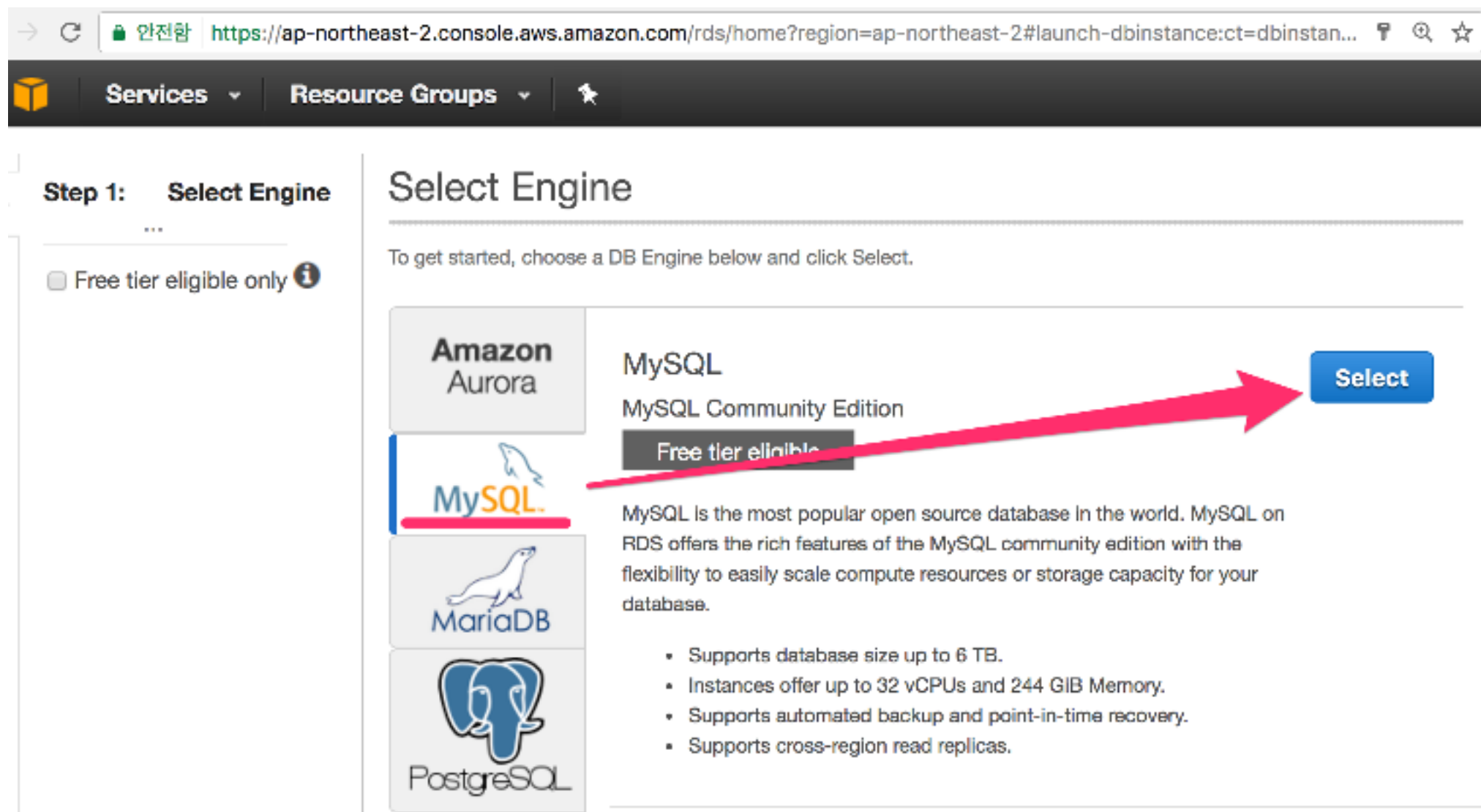


¹ 방화벽 서비스




² 본디 DB서버는 보안을 위해 외부 접속이 불가한 것이 좋습니다. 웹서비스 포트만 외부에 공개하는 것이 보안에 유리합니다. 혹은 특정 관리자IP대역만 접속을 허용하거나, VPN 혹은 SSH터널링을 활용합니다.

DB 엔진 선택 : MySQL

사용자 층이 가장 두터운 MySQL 선택



Dev/Test 선택 (Free tier)

 Services ▾ Resource Groups ▾   allieus ▾ Se

Step 1: [Select Engine](#)

Step 2: Production?

Step 3: Specify DB Details

Step 4: Configure Advanced Settings

Do you plan to use this database for production purposes?

Production

☐ Amazon Aurora

Recommended

MySQL-compatible, enterprise-class database at 1/10th the cost of commercial databases.

☐ MySQL

Use [Multi-AZ Deployment](#) and [Provisioned IOPS Storage](#) as defaults for high availability and fast, consistent performance.

Dev/Test

☒ MySQL

This instance is intended for use outside of production or under the [RDS Free Usage Tier](#).

Billing is based on [RDS pricing](#).

Cancel

Previous

Next Step

db.t2.micro/단일AZ³ 선택 (Free tier)

Services

Resource Groups

Step 1: [Select Engine](#)

Step 2: [Production?](#)

Step 3: **Specify DB Details**

Step 4: [Configure Advanced Settings](#)

Your current selection is eligible for the free tier.

[Learn More.](#)

Estimate your monthly costs for the DB Instance using the [RDS Instance Cost Calculator](#).

Specify DB Details

Free Tier

The Amazon RDS Free Tier provides a single db.t2.micro instance as well as up to 20 GB of storage, allowing new AWS customers to gain hands-on experience with Amazon RDS. Learn more about the RDS Free Tier and the instance restrictions [here](#).

☐ Only show options that are eligible for RDS Free Tier

Instance Specifications

DB Engine

mysql

License Model

general-public-license

DB Engine Version

MySQL 5.6.27

Review the **Known Issues/Limitations** to learn about potential compatibility issues with specific database versions.

DB Instance Class

db.t2.micro — 1 vCPU, 1 GiB RAM

Multi-AZ Deployment

No

Storage Type

General Purpose (SSD)

Allocated Storage^a

5 GB

³ 다중AZ 배포 시, 여러 가용영역에 동기식으로 복제하여 가용성 향상

AWS RDS/S3 서비스 이용하여 배포하기 / 장고 서비스 배포하기 / © AskDjango

11

DB 인스턴스 ID, DB_USER, DB_PASSWORD 설정

DB Engine Version: MySQL 5.6.27

Review the [Known Issues/Limitations](#) to learn about potential compatibility issues with specific database versions.

DB Instance Class: db.t2.micro — 1 vCPU, 1 GiB RAM

Multi-AZ Deployment: No

Storage Type: General Purpose (SSD)

Allocated Storage*: 5 GB

Provisioning less than 100 GB of General Purpose (SSD) storage for high throughput workloads could result in higher latencies upon exhaustion of the initial General Purpose (SSD) IO credit balance. [Click here](#) for more details.

Settings

DB Instance Identifier*: microdjango

Master Username*: microdjango

Master Password*:

Confirm Password*:

Retype the value you specified for Master Password.

* Required

Cancel Previous Next Step

생성한 **Securit Group**을 선택하고 동일한 **VPC**⁴ 지정

가용영역 (**Availability Zone**) 택일 및 **DB_NAME** 설정

Step 1: [Select Engine](#)
Step 2: [Production?](#)
Step 3: [Specify DB Details](#)
Step 4: **Configure Advanced Settings**

Configure Advanced Settings

Network & Security

VPC*
Subnet Group
Publicly Accessible
Availability Zone
VPC Security Group[s]

Database Options

Database Name
Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.
Database Port
DB Parameter Group
Option Group
Copy Tags To Snapshots ☐

Specify the TCP/IP port that the DB instance will use for application connections. The connection string of any application connecting to the DB instance must specify the port number of the DB instance. Both the security group applied to the DB instance and your company's firewalls must allow

⁴ 가상 네트워크 (Virtual Private Cloud)

Launch DB Instance - 생성을 시작해봅시다.

Database Port

3306

DB Parameter Group

default:mysql5.6

Option Group

default:mysql-5-6

Copy Tags To Snapshots

☐

Enable Encryption

No

Backup

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to detail [here](#).

Backup Retention Period

7 days

Backup Window

No Preference

Monitoring

Enable Enhanced Monitoring

No

Maintenance

Auto Minor Version Upgrade

Yes

Maintenance Window

No Preference

* Required

Cancel



Previous

Launch DB Instance

connection using an application connecting to the DB instance must specify the port number of the DB instance. Both the security group applied to the DB instance and your company's firewalls must allow connections to the port. [Learn More](#).



생성이 시작되었습니다.


 **Services** ▾ **Resource Groups** ▾ 

Step 1: [Select Engine](#)

Step 2: [Production?](#)

Step 3: [Specify DB Details](#)

Step 4: [Configure Advanced Settings](#)

 **Your DB Instance is being created.**

Note: Your instance may take a few minutes to launch.

Connecting to your DB Instance

Once Amazon RDS finishes provisioning your DB instance, you can use a SQL client application or utility to connect to the instance.

[Learn about connecting to your DB instance](#)

View Your DB Instances



생성 중입니다. 약 10분 정도 기다려보세요.

The screenshot shows the AWS RDS console interface. At the top, there's a navigation bar with 'Services', 'Resource Groups', and a user profile 'allieus' in 'Seoul'. The left sidebar lists various RDS features: RDS Dashboard, Instances, Clusters, Reserved Purchases, Snapshots, Parameter Groups, External Licenses, Option Groups, Subnet Groups, Events, Event Subscriptions, and Notifications. The main content area is titled 'Launch DB Instance' and includes buttons for 'Show Monitoring' and 'Instance Actions'. Below this is a filter section set to 'All Instances' and a search bar. A table displays one instance: 'microdjango' of type 'MySQL' with a status of 'creating'. Below the table, the 'Endpoint' is marked as 'Not available yet'. The 'Alarms and Recent Events' section shows 'No Recent Events'. The 'Monitoring' section displays metrics for CPU, Memory, and Storage, all with 'No Data' values. At the bottom, there are buttons for 'Instance Actions', 'Tags', and 'Logs'.

Engine	DB Instance	Status	CPU	Current Activity	Maintenance	Class	VPC	Multi-AZ
MySQL	microdjango	creating			None	db.t2.micro	vpc-ea10f183	No

	CURRENT VALUE	THRESHOLD	LAST HOUR	CURRENT VALUE	
CPU	No Data			Read IOPS	No Data
Memory	No Data			Write IOPS	No Data
Storage	No Data			Swap Usage	No Data

해당 DB Instance의 세부내역

The screenshot shows the AWS RDS console interface. On the left is the 'RDS Dashboard' sidebar with links to Instances, Clusters, Reserved Purchases, Snapshots, Parameter Groups, External Licenses, Option Groups, Subnet Groups, Events, Event Subscriptions, and Notifications. The main content area shows a list of DB instances. The instance 'microdjango' (MySQL engine, available status) is selected. The 'Instance Actions' dropdown menu is open, displaying the following options: See Details, Create Read Replica, Create Aurora Read Replica, Promote Read Replica, Take Snapshot, Restore to Point in Time, Migrate Latest Snapshot, Modify, Reboot, and Delete. A red arrow points from the 'Show Monitoring' button to the 'Instance Actions' dropdown. Below the instance list, the 'Endpoint' is shown as 'microdjango.cd93mgka6du6.ap-northeast-2.amazonaws.com'. The 'Alarms and Recent Events' section shows a table of events: 'Finished DB Instance backup' (Apr 21 10:46 AM), 'Backing up DB instance' (Apr 21 10:44 AM), and 'DB instance created' (Apr 21 10:43 AM). The 'Storage' section shows '4,540 MB'. The 'Performance' section shows metrics for Read IOPS (0/sec), Write IOPS (0.308/sec), and Swap Usage (0 MB).

DB연결정보를 확인해봅시다.

RDS Dashboard

Instances

Clusters

Reserved Purchases

Snapshots

Parameter Groups

External Licenses

Option Groups

Subnet Groups

Events

Event Subscriptions

Notifications

DB Instances > microdjango

DetailsRecent Events & Logs

Endpoint: [microdjango.cd93mpka6du6.ap-northeast-2.rds.amazonaws.com:3306](#) (**authorized**) ⓘ

Configuration Details

ARN

arn:aws:rds:ap-northeast-2:095307090881:db:microdjango

Engine

MySQL 5.6.27

License Model

General Public License

Created Time

April 21, 2017 at 10:43:14 AM UTC+9

DB Name

microdjango

Username

microdjango

Option Group

default:mysql-5-6 (**in-sync**)

Parameter Group

default:mysql5.6 (**in-sync**)

Copy Tags To Snapshots

No

Resource ID

db-ATOSZMIMJ4GCCXIRKEKTZIQNGA

Security and Network

Availability Zone

ap-northeast-2a

VPC

vpc-ea10f183

Subnet Group

default (**Complete**)

Subnets

subnet-d1a849b8
subnet-3fcdee75

Security Groups

rds-mysql-security-group (sg-865ddaee)
(active)

Publicly Accessible

Yes

Endpoint

microdjango.cd93mpka6du6.ap-northeast-2.rds.amazonaws.com

Port

3306

Certificate Authority

rds-ca-2015 (**Mar 5, 2020**)

Instance and IOPS

Instance Class

db.t2.micro ⓘ

Storage Type

General Purpose (SSD)

IOPS

disabled

Storage

5 GB

DB_HOST

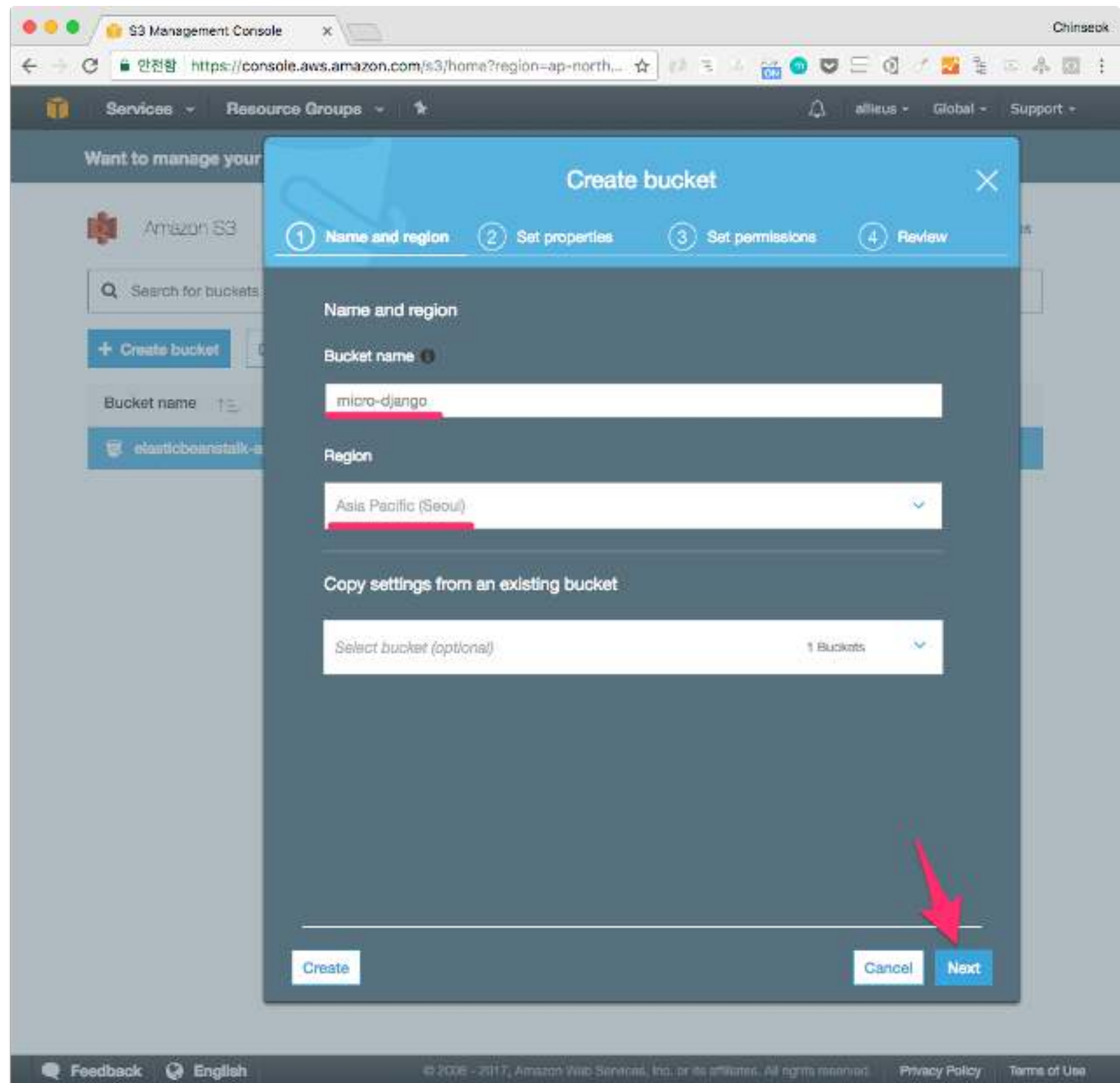
AWS S3

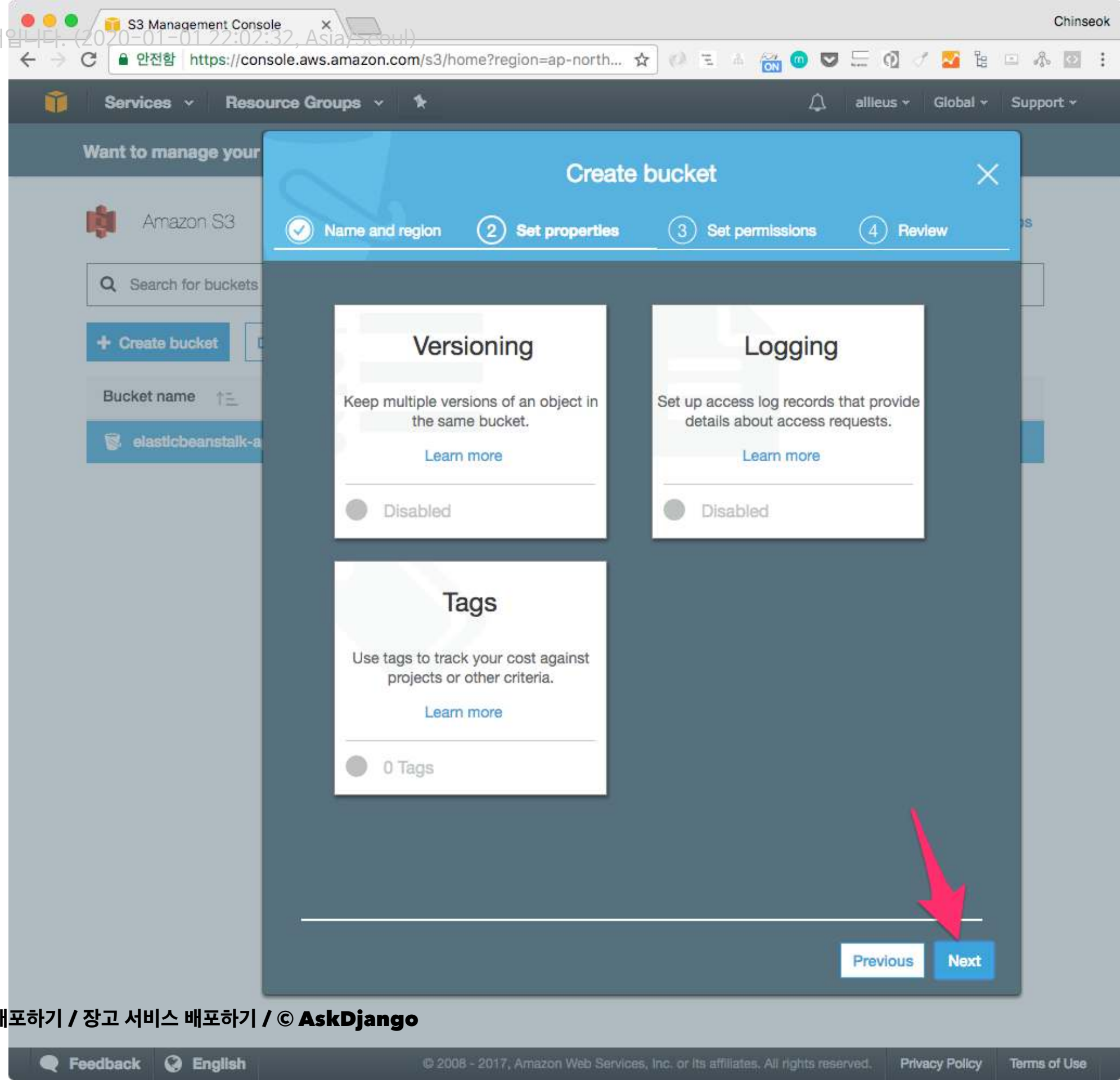
Simple Storage Service

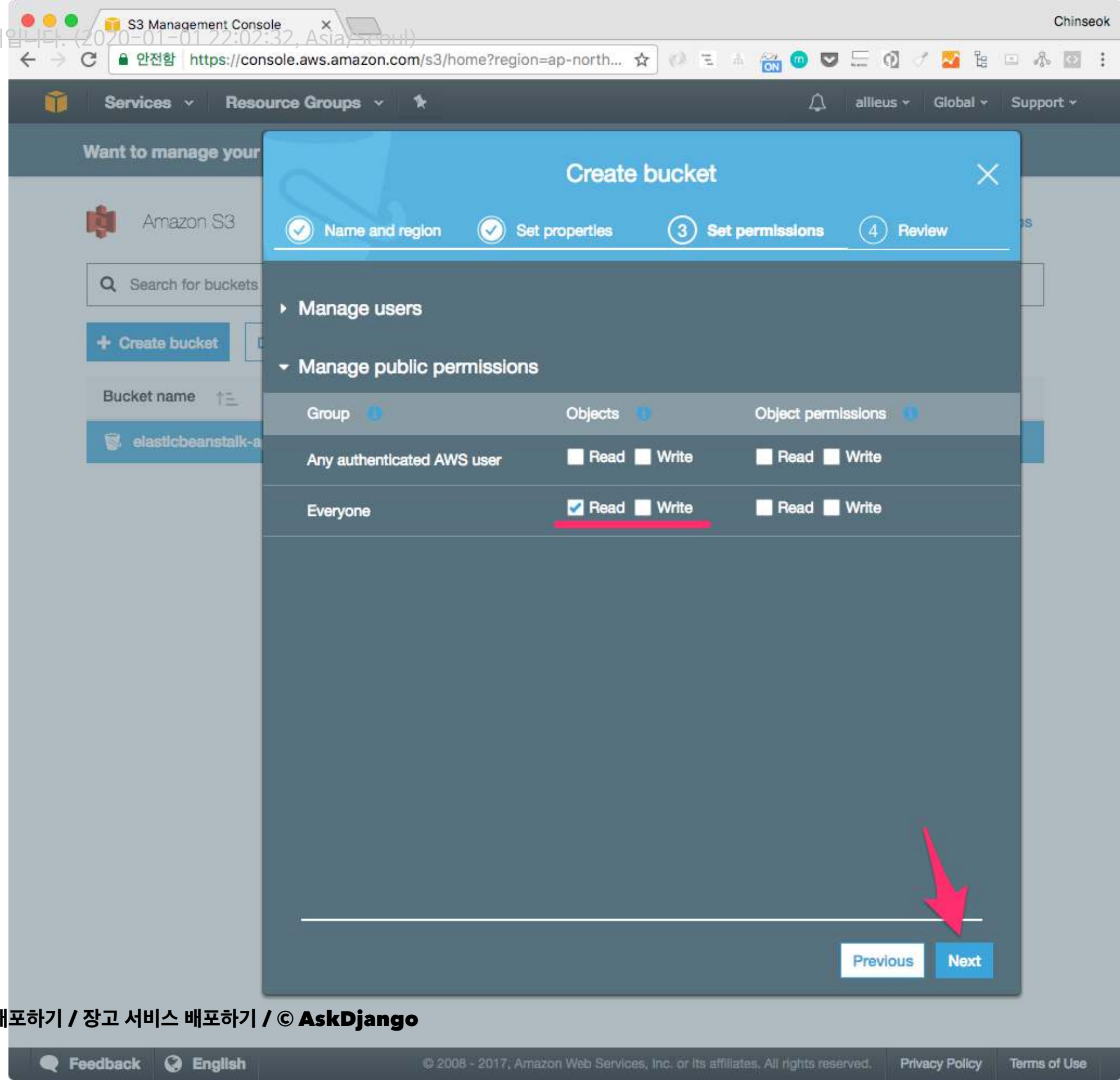
S3 **#doc**

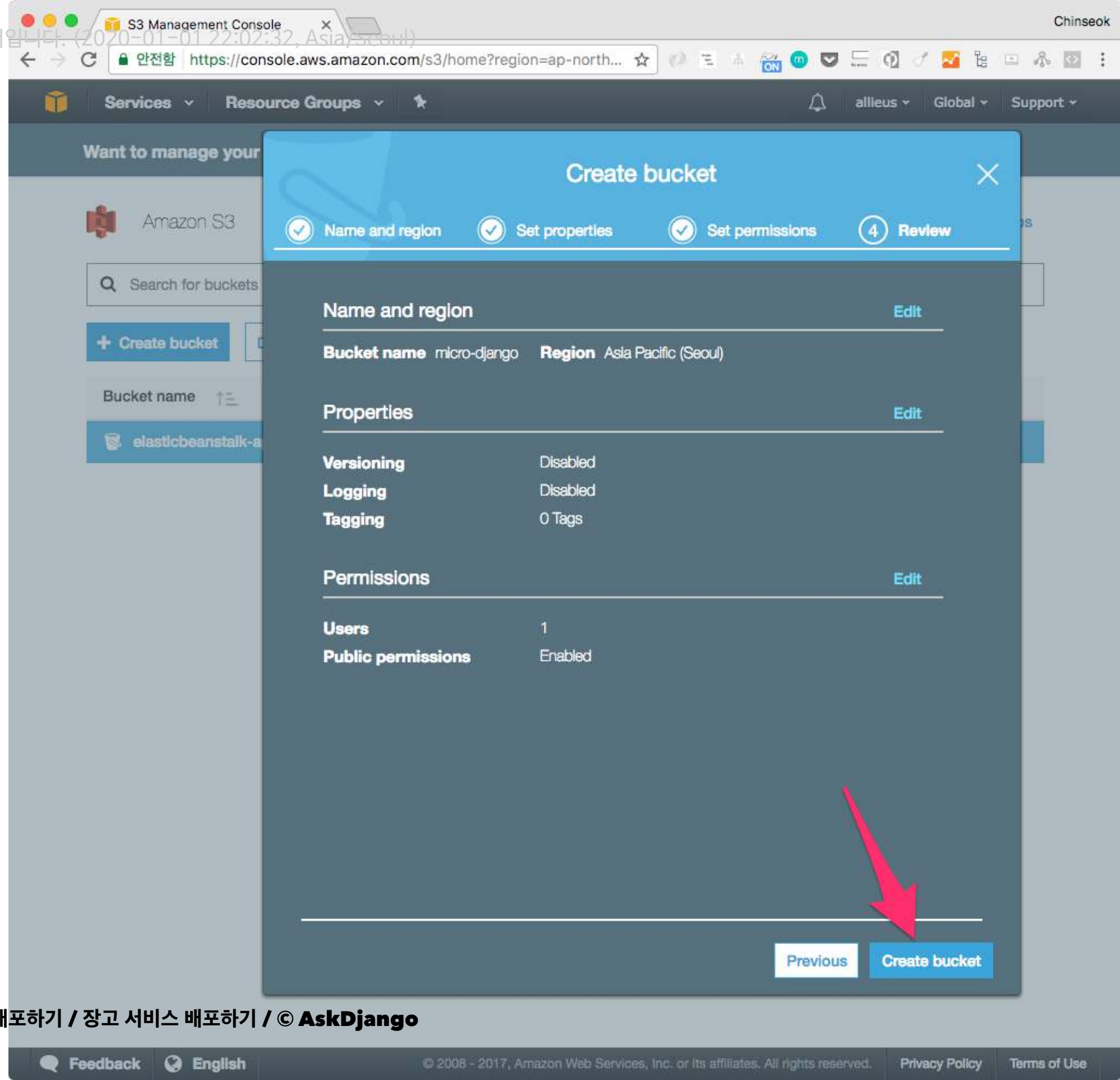
- 인터넷용 스토리지 서비스
- 높은 확장성과 신뢰성
- 전 세계적으로 99.9999999999%의 내구성을 제공

Create Bucket



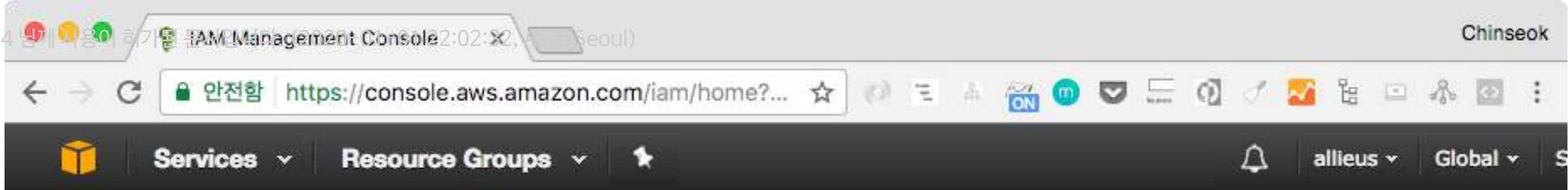




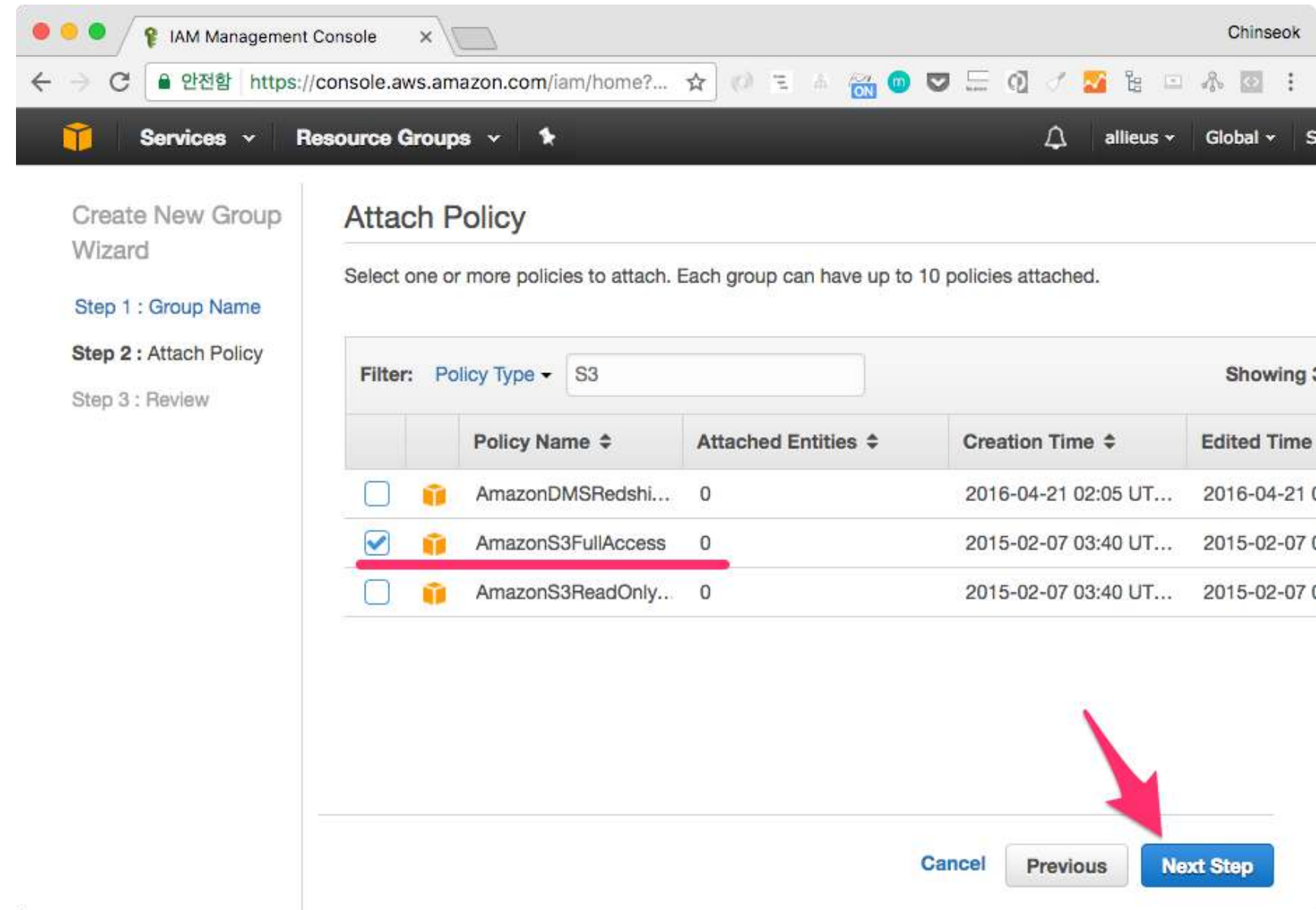


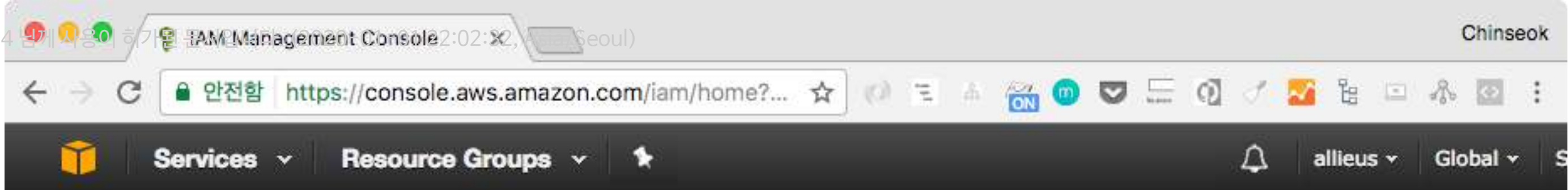
EB Webtier를 위한 IAM 유저 생성

지금은 AmazonS3FullAccess 권한만 필요



지금은 AmazonS3FullAccess 권한 이면 충분





Create New Group Wizard

Step 1 : Group Name

Step 2 : Attach Policy

Step 3 : Review

Review

Review the following information, then click **Create Group** to proceed.

Group Name eb-web

Policies arn:aws:iam::aws:policy/AmazonS3FullAccess

Cancel

Previous

Create Group

새 유저를 생성

The screenshot shows the AWS IAM Management Console interface. At the top, there's a navigation bar with 'Services' and 'Resource Groups' dropdowns, and a user profile 'allieus'. Below this, a progress bar indicates four steps: 1. Details (active), 2. Permissions, 3. Review, and 4. Complete. The main heading is 'Add user', followed by 'Set user details'. A subtext explains that multiple users can be added at once. The 'User name*' field contains 'eb-web-user'. Below it is a link to 'Add another user'. The 'Select AWS access type' section explains that access keys and passwords are provided in the last step. Under 'Access type*', 'Programmatic access' is selected with a checkbox, and 'AWS Management Console access' is unselected. A large red arrow points to the 'Next: Permissions' button at the bottom right. The footer includes 'Feedback', 'English', copyright information, and a 'Privacy Policy' link.

IAM Management Console

Chinseok

Services Resource Groups

allieus Global

Add user

1 Details 2 Permissions 3 Review 4 Complete

Set user details

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

User name* eb-web-user

[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

Feedback English

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy

방금 생성한 eb-web 그룹에 추가

Add user

1

Details

2

Permissions

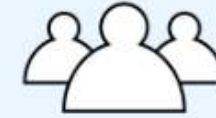
3

Review

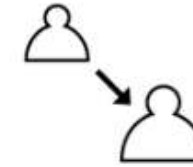
4

Complete

Set permissions for eb-web-user



Add user to group

Copy permissions from
existing userAttach existing policies
directly

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

Create group

Refresh

Q Search

Showing 2 results

Group ▾

Attached policies

☐ eb-deploy

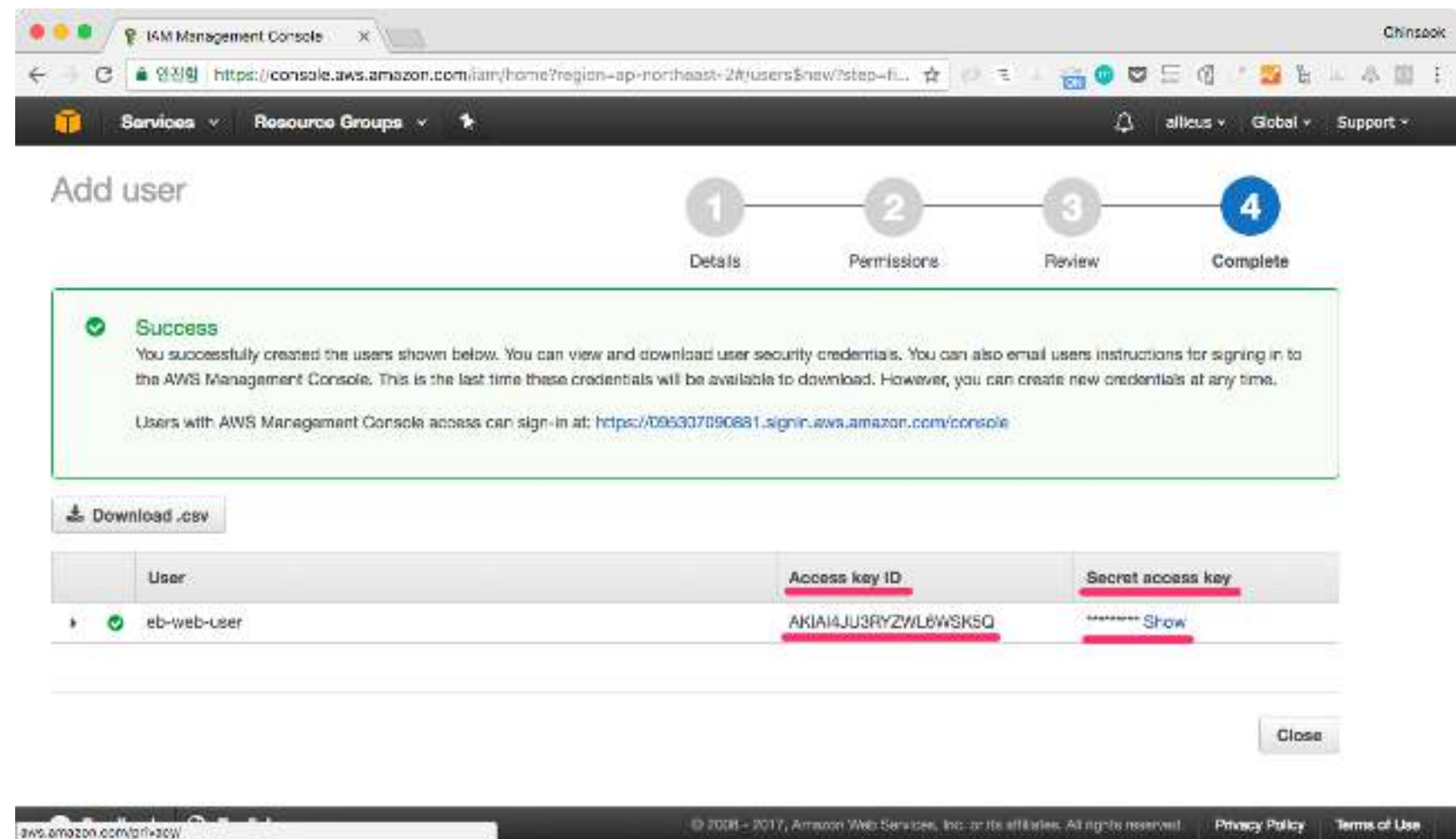
AWSElasticBeanstalkFullAccess

☒ eb-web

AmazonS3FullAccess

생성 완료

ACCESS/SECRET KEY를 갈무리해주세요.



배포 준비

필요한 설정 분기

- 구동환경별 requirements.txt 분기
 - 개발용 (reqs/dev.txt)
 - AWS EB 배포용 (reqs/prod_aws_eb.txt)
- 구동환경별 settings 분기
 - 개발용 (settings.dev)
 - AWS EB 배포용 (settings.prod_aws_eb)

requirements.txt 분기

AWS EB에서만 현재 필요한 패키지

- django-storages : static/media 저장/서빙을 S3에서 처리
- boto3 : AWS SDK
- PyMySQL : Pure Python MySQL Client

RDS를 위한 **settings** 분기

```
# 프로젝트.settings.prod_aws_eb
import os

import pymysql
pymysql.install_as_MySQLdb() # pymysql이 MySQLdb처럼 동작토록 세팅

DATABASES = {
    'default': {
        'ENGINE': os.environ.get('DB_ENGINE', 'django.db.backends.mysql'),
        'HOST': os.environ['DB_HOST'],
        'USER': os.environ['DB_USER'],
        'PASSWORD': os.environ['DB_PASSWORD'],
        'NAME': os.environ['DB_NAME'],
        'PORT': os.environ['DB_PORT'],
    },
}
```

S3를 위한 settings 분기

static/media 파일 추가/삭제

```
# 프로젝트.settings.prod_aws_eb
import os

INSTALLED_APPS += ['storages'] # django-storages 앱 의존성 추가

# 기본 static/media 저장소를 django-storages로 변경
STATICFILES_STORAGE = '프로젝트.storages.StaticS3Boto3Storage'
DEFAULT_FILE_STORAGE = '프로젝트.storages.MediaS3Boto3Storage'

# S3 파일 관리에 필요한 최소 설정
# 소스코드에 설정정보를 남기지마세요. 환경변수를 통한 설정 추천
AWS_ACCESS_KEY_ID = os.environ['AWS_ACCESS_KEY_ID'] # 필수 지정
AWS_SECRET_ACCESS_KEY = os.environ['AWS_SECRET_ACCESS_KEY'] # 필수 지정
AWS_STORAGE_BUCKET_NAME = os.environ['AWS_STORAGE_BUCKET_NAME'] # 필수 지정

AWS_S3_REGION_NAME = os.environ.get('AWS_S3_REGION_NAME', 'ap-northeast-2')
```

Custom Storage

```
# 프로젝트/storages.py

from storages.backends.s3boto3 import S3Boto3Storage

class StaticS3Boto3Storage(S3Boto3Storage):
    location = 'static' # bucket 업로드 prefix 지정

class MediaS3Boto3Storage(S3Boto3Storage):
    location = 'media' # bucket 업로드 prefix 지정
```

settings에 위 Storage를 지정하면

- StaticS3Boto3Storage
 - python3 manage.py collectstatic 명령 시에 S3내 bucket/static 경로에 모든 static 파일 복사
 - 모든 static 서빙 URL이 S3 bucket/static으로 지정
- MediaS3Boto3Storage
 - 모든 파일 업로드가 bucket/media 경로에 저장
 - 모든 media 서빙 URL이 S3 bucket/media로 지정

AWS S3 Region #ref

Region Name	Region	Region Name	Region
US East (N. Virginia)	us-east-1	US East (Ohio)	us-east-2
US West (N. California)	us-west-1	US West (Oregon)	us-west-2
Canada (Central)	ca-central-1	Asia Pacific (Mumbai)	ap-south-1
Asia Pacific (Seoul)	ap-northeast-2	Asia Pacific (Singapore)	ap-southeast-1
Asia Pacific (Sydney)	ap-southeast-2	Asia Pacific (Tokyo)	ap-northeast-1
EU (Frankfurt)	eu-central-1	EU (Ireland)	eu-west-1
EU (London)	eu-west-2	South America (São Paulo)	sa-east-1

.ebextensions/options.config

```
packages:
```

```
  yum:
```

```
    freetype-devel: []
```

```
    libjpeg-turbo-devel: []
```

```
    libpng-devel: []
```

```
option_settings:
```

```
  aws:elasticbeanstalk:container:python:
```

```
    WSGIPath: 프로젝트/wsgi.py
```

- pillow 라이브러리 설치를 위해, 필요한 패키지를 yum 패키지관리자를 통해 설치
- static 서빙은 이제 EB측 아파치 웹서버가 아니라, S3에서 직접 서빙
- 환경변수는 EB Environment 측에 직접 세팅
- container_commands는 모두 제거하고, 독립적으로 수행하겠습니다. (migrate, collectstatic, chmod_sqlite, create_superuser)

로컬에서 **RDS/S3**관련 명령을 내리기 전에, 필요한 로컬 환경변수 세팅

맥/리눅스를 쓰신다면?

```
셸> export DJANGO_SETTINGS_MODULE=helloeb2.settings.prod_aws_eb
```

```
셸> export DB_HOST=''
```

```
셸> export DB_USER=''
```

```
셸> export DB_PASSWORD=''
```

```
셸> export DB_PORT=''
```

```
셸> export DB_NAME=''
```

```
셸> export AWS_ACCESS_KEY_ID='' # IAM ACCESS KEY를 지정
```

```
셸> export AWS_SECRET_ACCESS_KEY='' # IAM SECRET KEY를 지정
```

```
셸> export AWS_STORAGE_BUCKET_NAME='' # 생성한 S3 Bucket Name을 지정
```

```
셸> export AWS_S3_REGION_NAME='ap-northeast-2' # Seoul
```

필히 이어서 터미널 실행

터미널을 종료하면 재지정 (혹시나 쉘 설정파일에 넣어둔다면, 관리에 유의할 것)

윈도우 / 명령 프롬프트를 쓰신다면?

```
셸> set DJANGO_SETTINGS_MODULE=프로젝트.settings.prod_aws_eb
```

```
셸> set DB_HOST=' '
```

```
셸> set DB_USER=' '
```

```
셸> set DB_PASSWORD=' '
```

```
셸> set DB_PORT=' '
```

```
셸> set DB_NAME=' '
```

```
셸> set AWS_ACCESS_KEY_ID=' '
```

```
셸> set AWS_SECRET_ACCESS_KEY=' '
```

```
셸> set AWS_STORAGE_BUCKET_NAME=' '
```

```
셸> set AWS_S3_REGION_NAME='ap-northeast-2'
```

로컬에서 RDS/S3에 대한 선행작업

migrate 및 superuser 생성

```
python3 manage.py migrate --settings=프로젝트.settings.prod_aws_eb
```

```
python3 manage.py createsuperuser --settings=프로젝트.settings.prod_aws_eb
```

S3 bucket/name으로 static files 복사

```
python3 manage.py collectstatic --settings=프로젝트.settings.prod_aws_eb
```

이제, 로컬에서 "프로젝트.settings.prod_aws_eb" settings를 통해 서버를 실행시켜보세요. 잘 뜨나요? :D

배포 **EB**환경에 환경변수 세팅

혹은 **EB**웹콘솔 / **Configuration**에서 세팅

```
셸> eb setenv \  
    DJANGO_SETTINGS_MODULE=프로젝트.settings.prod_aws_eb \  
    AWS_ACCESS_KEY_ID=' ' \  
    AWS_SECRET_ACCESS_KEY=' ' \  
    DB_HOST=' ' \  
    DB_USER=' ' \  
    DB_PASSWORD=' ' \  
    DB_PORT=' ' \  
    DB_NAME=' ' \  
    AWS_STORAGE_BUCKET_NAME=' ' \  
    AWS_S3_REGION_NAME='ap-northeast-2'
```

Elastic Beanstalk 배포

- 필요할 때마다 직접 migrate/collectstatic
- 그리고, 장고 애플리케이션 배포

셸> eb deploy

*Life is short,
use Python3/Django.*