Tutorial for deploying the “Blob or Not” application through AWS Elastic Beanstalk

Last updated on March 2021

Overview

In order to deploy the “Blob or Not”application via AWS, we need to:

1) create an EB environment

2) create a RDS database instance

3) upload images to S3

4) create an IAM user to access all services

5) preparing the Flask codes.

Content

1. AWS Account

You can use the following link to sign up an AWS account:

https://portal.aws.amazon.com/billing/signup#/start

2. Elastic Beanstalk

Elastic Beanstalk (EB) is the place where we deploy our application, so let’s first create an EB environment.

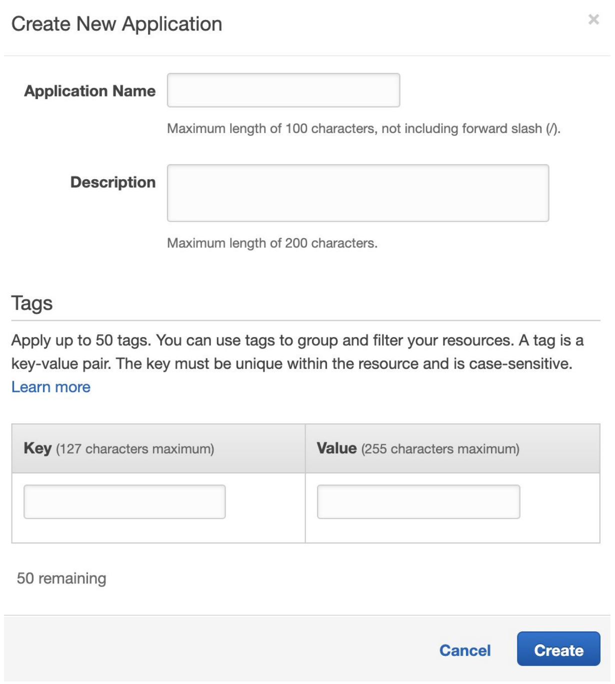
1. Go to AWS Management Console
2. Click **Elastic Beanstalk**
3. Click **Create New Application**
4. type the Application Name

If this is your first EB application and are given the platform dropdown option:

5. choose python for the platform

Otherwise:

- Click ‘Create’



Create New Application

Application Name

Maximum length of 100 characters, not including forward slash (/).

Description

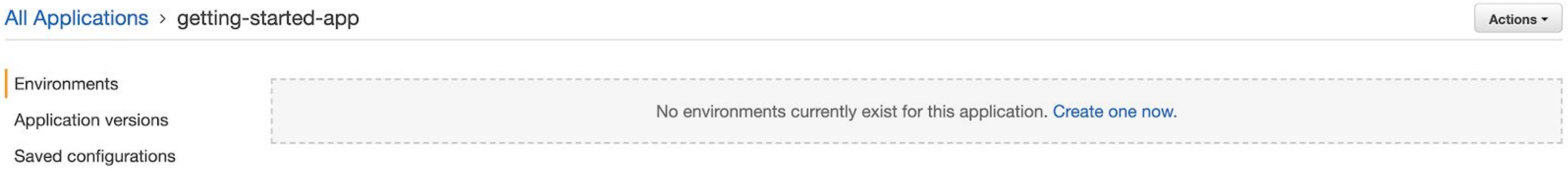
Maximum length of 200 characters.

Tags

Apply up to 50 tags. You can use tags to group and filter your resources. A tag is a key-value pair. The key must be unique within the resource and is case-sensitive. Learn more

Key (127 characters maximum)	**Value** (255 characters maximum)

50 remaining



No environments currently exist for this application. Create one now.

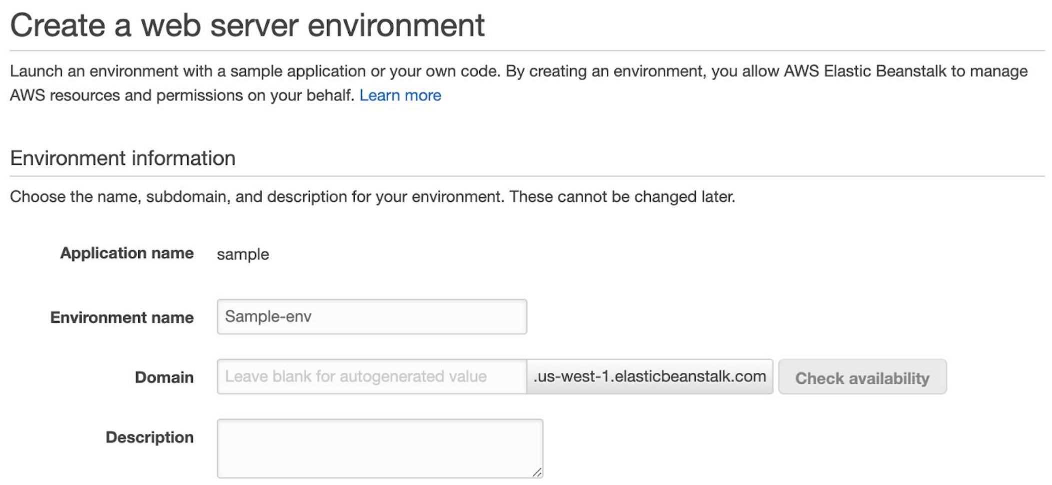
NOTE: BRAND NEW AWS ACCOUNTS MIGHT NOT SEE THIS OPTION

Click **create on now** to create an environment

Select “Web Server Environment”

Set environment name and domain name, then create using a preconfigured Python environment (not Docker, just deploy their sample now, we can upload our code later)

Please remember your **Environment name**



Create a web server environment

Launch an environment with a sample application or your own code. By creating an environment, you allow AWS Elastic Beanstalk to manage AWS resources and permissions on your behalf. Learn more

Environment information

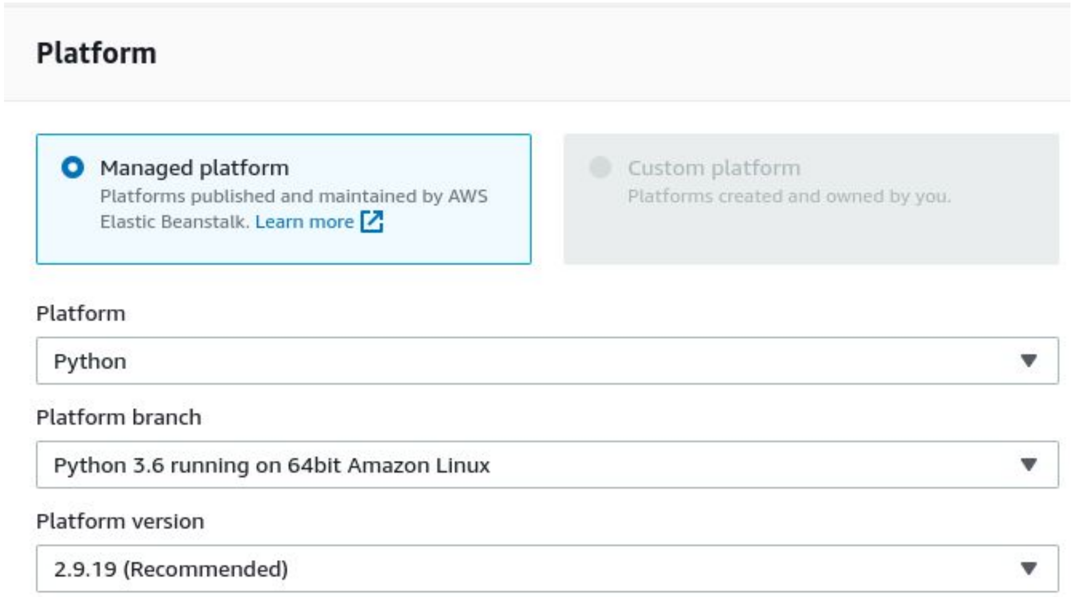
Choose the name, subdomain, and description for your environment. These cannot be changed later.

Application name	sample
Environment name	Sample-env
Domain	Leave blank for autogenerated value .us-west-1.elasticbeanstalk.com Check availability
Description	

NOTE: BRAND NEW AWS ACCOUNTS MIGHT
NOT SEE THIS OPTION

Use a Managed platform. Select:

- Python platform
- **Python 3.6 running on 64bit Amazon Linux branch**
- **Platform version 2.9.19**



Platform

Managed platform
Platforms published and maintained by AWS Elastic Beanstalk. Learn more

Custom platform
Platforms created and owned by you.

Platform: Python

Platform branch: Python 3.6 running on 64bit Amazon Linux

Platform version: 2.9.19 (Recommended)

2. RDS

Ok, we have created our EB environment, it will take some time to complete, so let’s create our database instance.

We will use RDS as database to store our annotation records.

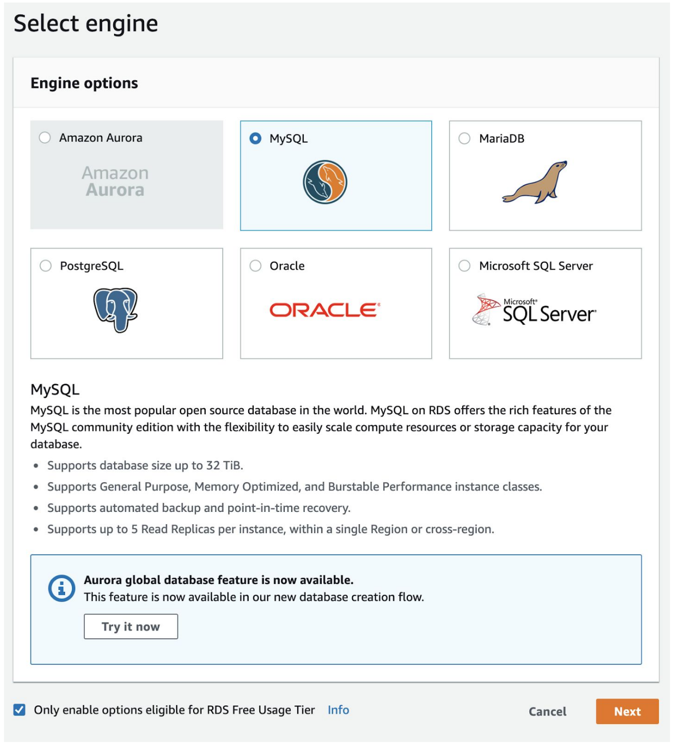
RDS

In the AWS Management Console, Go to RDS

Click **create database**

Select **MySQL**

Under Templates, select only enable options for **Free Usage Tier**



MySQL

MySQL is the most popular open source database in the world. MySQL on RDS offers the rich features of the MySQL community edition with the flexibility to easily scale compute resources or storage capacity for your database.

- Supports database size up to 32 TiB.
- Supports General Purpose, Memory Optimized, and Burstable Performance instance classes.
- Supports automated backup and point-in-time recovery.
- Supports up to 5 Read Replicas per instance, within a single Region or cross-region.

Aurora global database feature is now available.
This feature is now available in our new database creation flow.

[✓] Only enable options eligible for RDS Free Usage Tier

RDS

Set **DB identifier**, **username**, and **password**

Under Additional configuration, set **Initial database name**

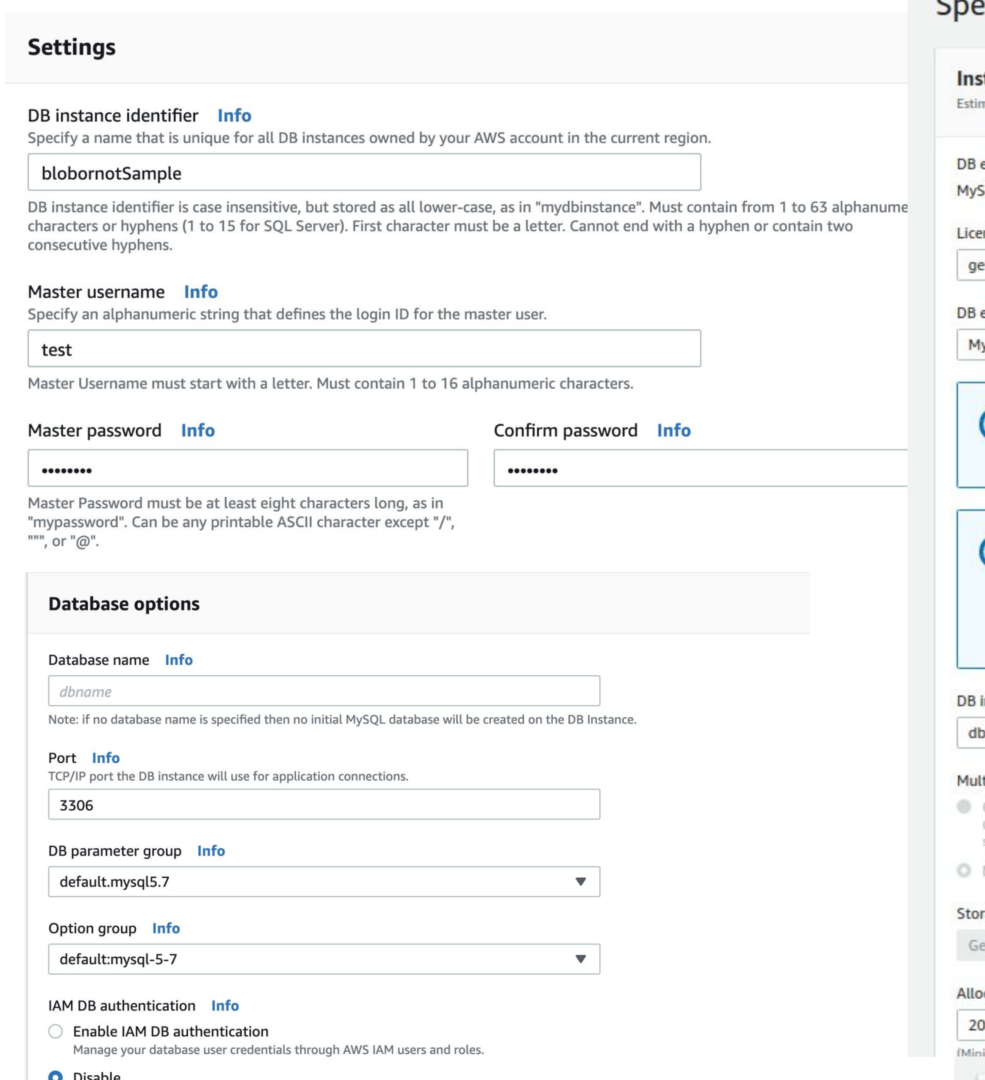
Leave everything else as default

Create!

Please remember the

Master username, Master password, and Database name

They will be used later



Settings

DB instance identifier Info
Specify a name that is unique for all DB instances owned by your AWS account in the current region.

blobornotSample

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance". Must contain from 1 to 63 alphanume characters or hyphens (1 to 15 for SQL Server). First character must be a letter. Cannot end with a hyphen or contain two consecutive hyphens.

Master username Info
Specify an alphanumeric string that defines the login ID for the master user.

test

Master Username must start with a letter. Must contain 1 to 16 alphanumeric characters.

Master password Info | Confirm password Info

•••••••• | ••••••••

Master Password must be at least eight characters long, as in "mypassword". Can be any printable ASCII character except "/", """, or "@".

Database options

Database name Info

dbname

Note: if no database name is specified then no initial MySQL database will be created on the DB Instance.

Port Info
TCP/IP port the DB instance will use for application connections.

3306

DB parameter group Info

default.mysql5.7

Option group Info

default:mysql-5-7

IAM DB authentication Info

Enable IAM DB authentication
Manage your database user credentials through AWS IAM users and roles.

Disable

Security Group

Ok, we have created our database instance. We need to set a security group to this instance so that we can access it from our EB application.

Create security group

Go to AWS Management Console

Go to **EC2**

On the left, click **Security Groups**

Click create security group

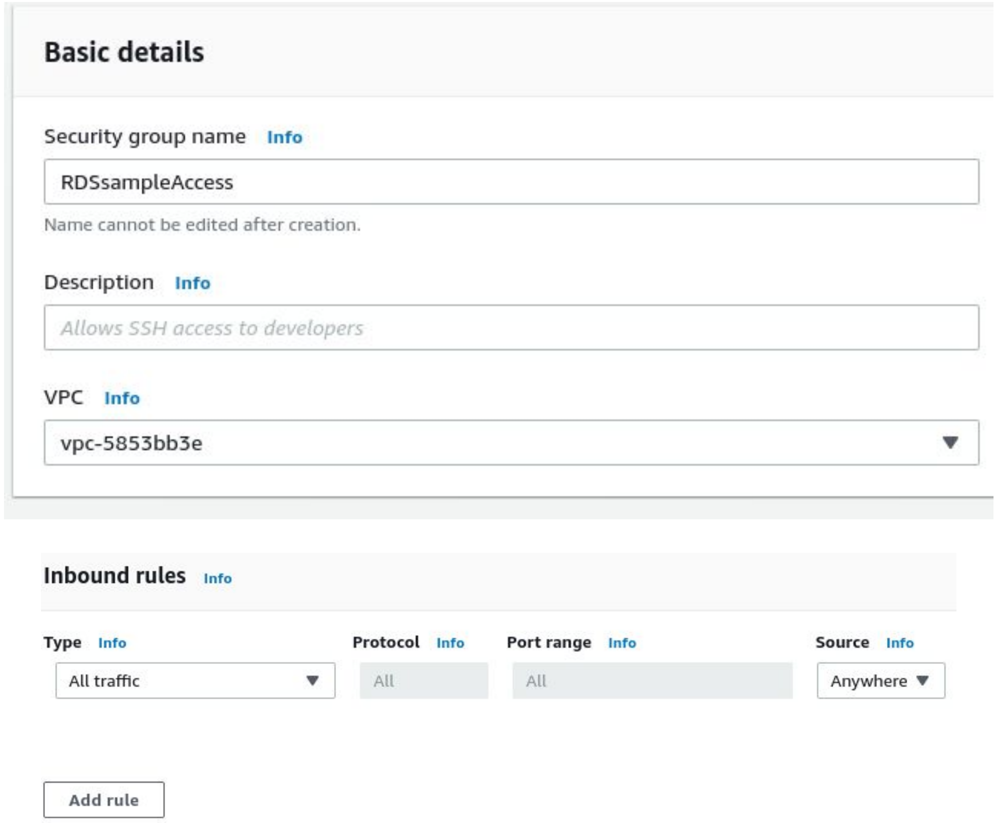
Set **group name**

Add rule to allow **all traffic**

Set Source to **anywhere**

Create!

Please remember the **group security name**



Basic details

Security group name Info

RDSsampleAccess

Name cannot be edited after creation.

Description Info

Allows SSH access to developers

VPC Info

vpc-5853bb3e

Inbound rules Info

Type	Protocol	Port range	Source
All traffic	All	All	Anywhere

Add security group to RDS

Go back to RDS

Select your DB instance

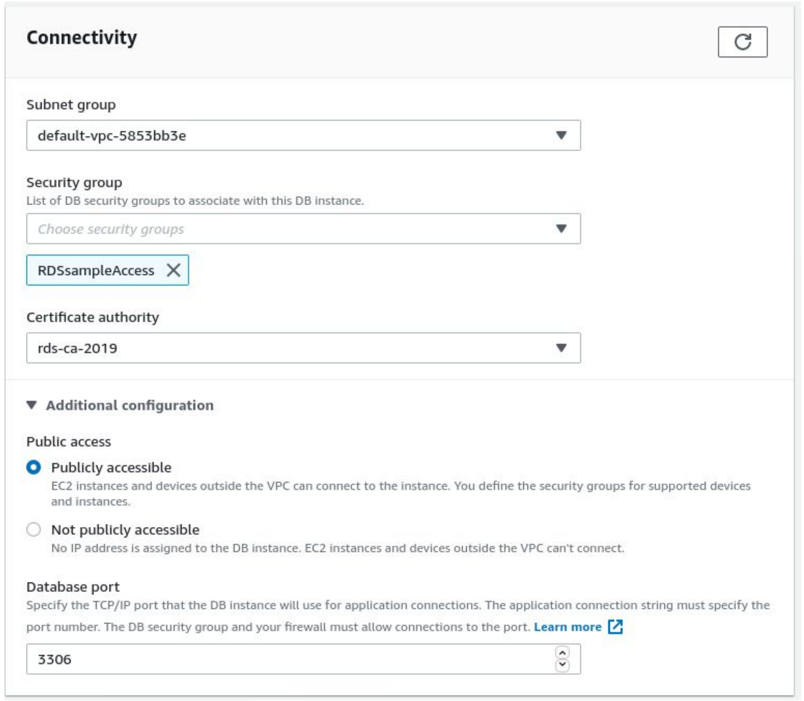
Click modify

Add the **security group** you just created

Set **public accessibility to Yes**
- this shouldn't be necessary. If we were to change our config to use the DB environment variables set in EB and we figured out how to create the db tables and back them up w/ out running scripts locally, we could keep this off. Perhaps a cron-job for backups (see cron-leader-only here: https://aws.amazon.com/premiumsupport/knowledge-center/cron-job-elastic-beanstalk/).

Click Continue

Select apply immediately and click Modify DB Instance!



Connectivity

Subnet group
default-vpc-5853bb3e

Security group
List of DB security groups to associate with this DB instance.
Choose security groups
RDSsampleAccess

Certificate authority
rds-ca-2019

▼ Additional configuration

Public access
● Publicly accessible
EC2 instances and devices outside the VPC can connect to the instance. You define the security groups for supported devices and instances.
○ Not publicly accessible
No IP address is assigned to the DB instance. EC2 instances and devices outside the VPC can't connect.

Database port
Specify the TCP/IP port that the DB instance will use for application connections. The application connection string must specify the port number. The DB security group and your firewall must allow connections to the port. Learn more
3306

Add security group to EB

Open the Elastic Beanstalk console

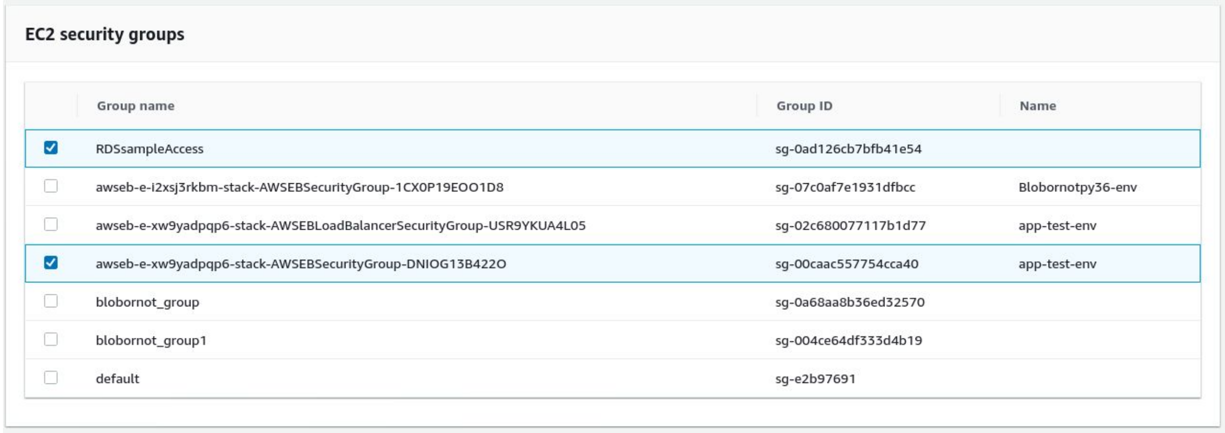
Choose the environment you just created

Choose **configuration**

Under **Instances**, click Edit

Under **EC2 security group**, check the security group you just created

Apply!



EC2 security groups

	Group name	Group ID	Name
✓	RDSsampleAccess	sg-0ad126cb7bfb41e54	
☐	awseb-e-i2xsj3rkbm-stack-AWSEBSecurityGroup-1CX0P19EOO1D8	sg-07c0af7e1931dfbcc	Blobornotpy36-env
☐	awseb-e-xw9yadpqp6-stack-AWSEBLoadBalancerSecurityGroup-USR9YKUA4L05	sg-02c680077117b1d77	app-test-env
✓	awseb-e-xw9yadpqp6-stack-AWSEBSecurityGroup-DNIOG13B422O	sg-00caac557754cca40	app-test-env
☐	blobornot_group	sg-0a68aa8b36ed32570	
☐	blobornot_group1	sg-004ce64df333d4b19	
☐	default	sg-e2b97691	

Connect EB to RDS

Open Elastic Beanstalk console

Choose the environment you just created

Go to configuration, under **software**, click Edit

In the **Environment properties** section, define the variables that your application reads to construct a connection string.

- **RDS_HOSTNAME** – The hostname of the DB instance.
  Amazon RDS console label – **Endpoint** (this is the hostname)
- **RDS_PORT** – The port on which the DB instance accepts connections. The default value varies among DB engines.
  Amazon RDS console label – **Port**
- **RDS_DB_NAME** – The database name.
  Amazon RDS console label – **DB Name**
- **RDS_USERNAME** – The user name that you configured for your database.
  Amazon RDS console label – **Username**
- **RDS_PASSWORD** – The password that you configured for your database.

Apply!

4. S3

We have created our EB environment and RDS database and connected them together.

Now, we need to create a S3 bucket. This will be used to upload annotation images and our application will read images from it.

Go to the S3 console

Click Create bucket

Set the **bucket name**

Choose the same **AWS region** as your EB environment.

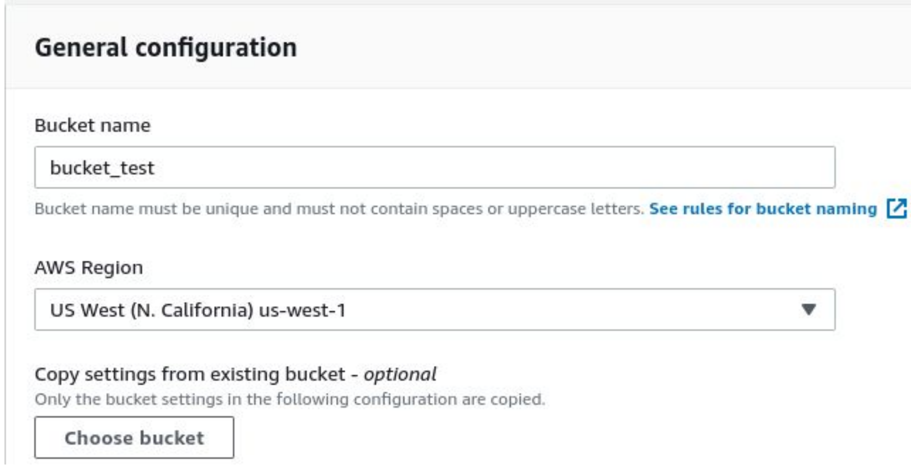
Click Next through to Permissions

Permissions: Unclick block all public access

Remain other settings default, Create!

All images should be upload to this bucket for annotation

Please remember the **bucket name**



General configuration

Bucket name

bucket_test

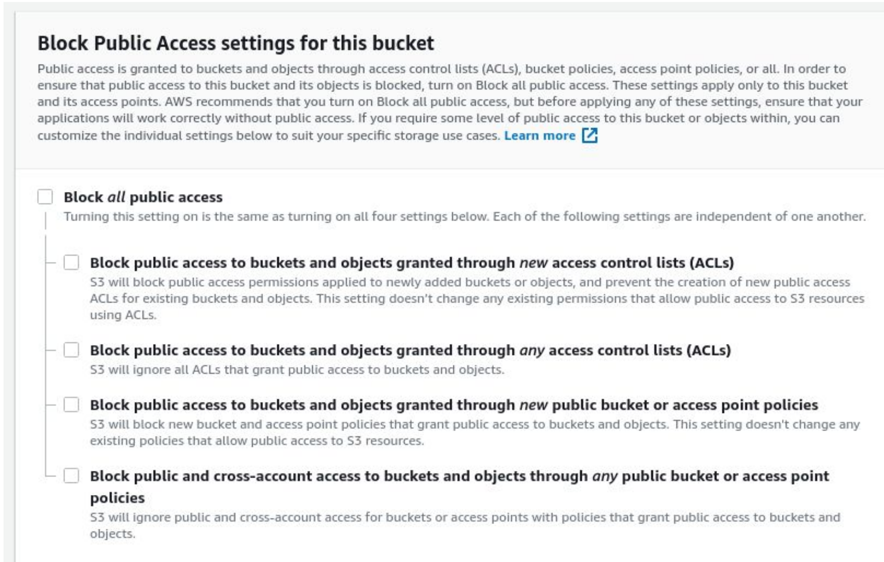
Bucket name must be unique and must not contain spaces or uppercase letters. See rules for bucket naming

AWS Region

US West (N. California) us-west-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.



Block Public Access settings for this bucket

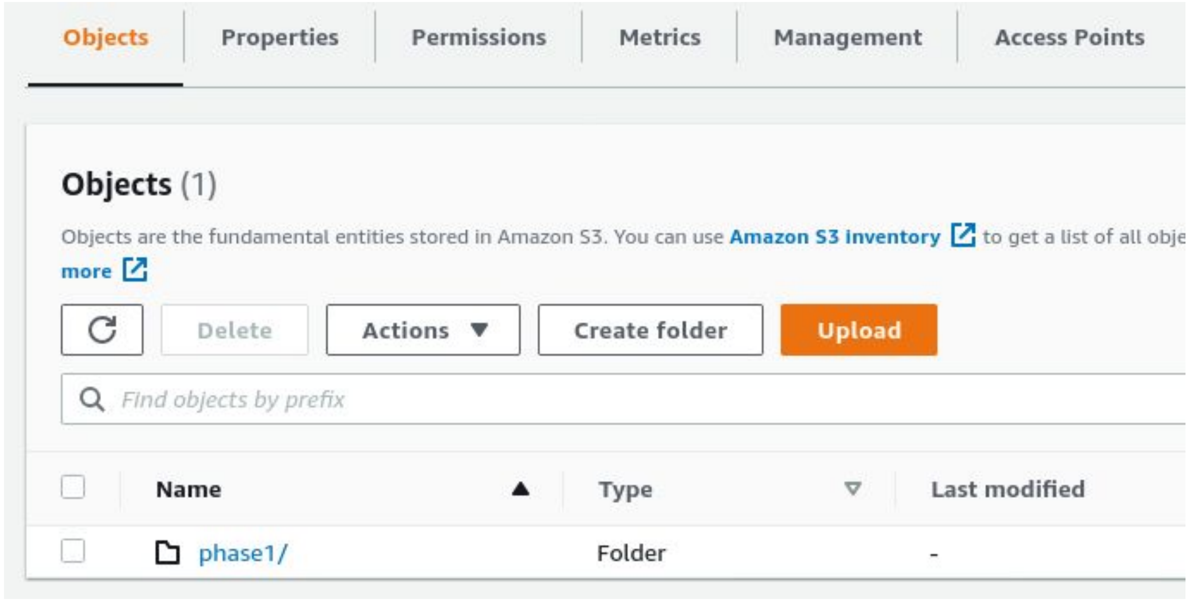
Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. 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 this bucket 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 or access point policies**
  S3 will block new bucket and access point 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 or access point policies**
  S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Bucket

Click into the bucket, here you can create folders and subfolders



Objects (1)

Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all obje
more

Name	Type	Last modified
phase1/	Folder	-

5. IAM

We have set up all services we need. Now we need to create a user who has access to all these services.

Add security user

Go to IAM

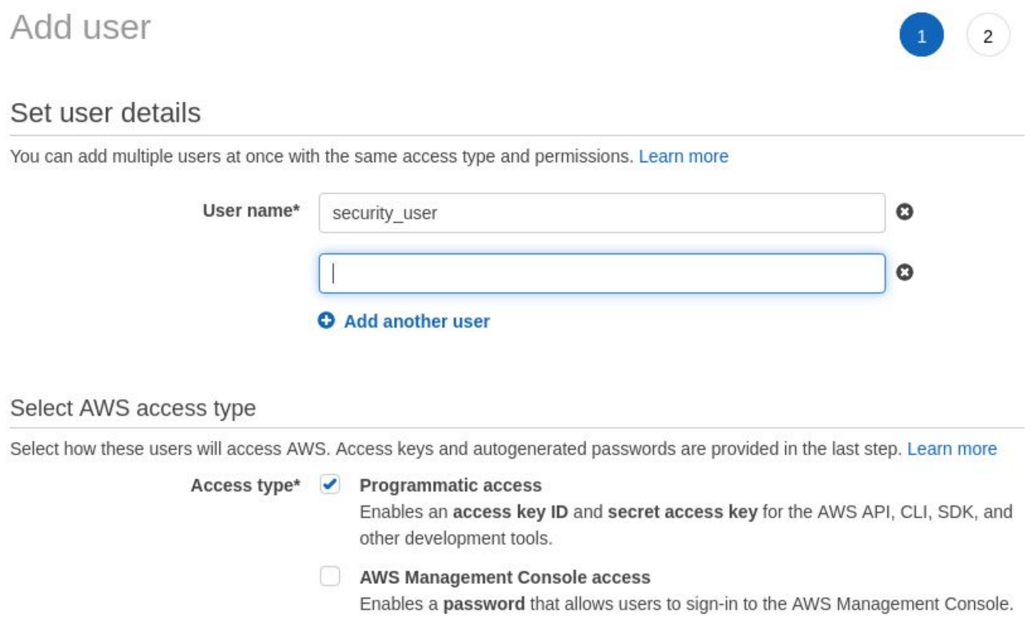
On the left, click **Users**

Click Add user

Set username

Select “programmatic access”

Next: Permissions!



Add user

1 2

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* security_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.

Add permission for the user

Then, create a group, set **group name**

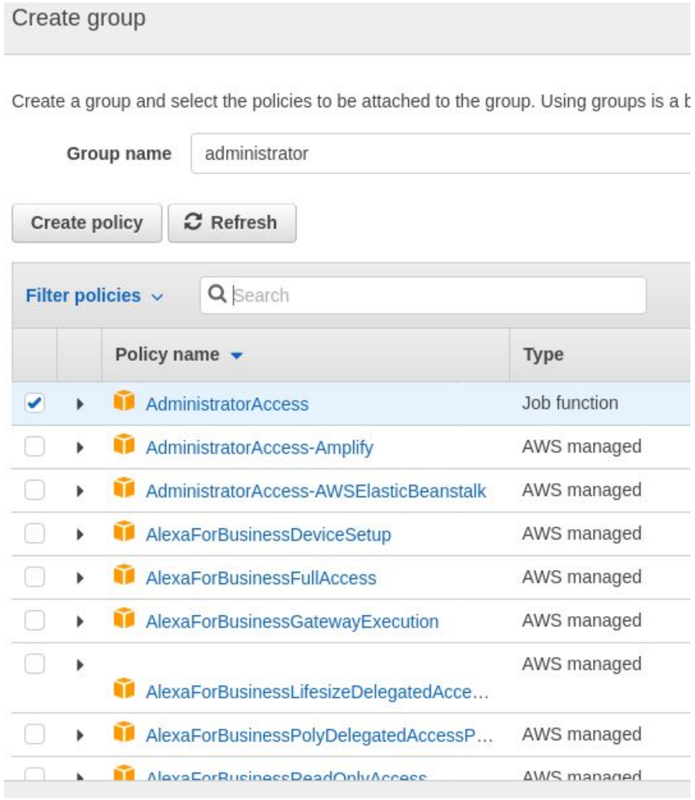
Search for “administrator”,check “AdministratorAccess”

Search for “EC2”, check “AmazonEC2FullAccess”

Search for “S3”, check “AmazonS3FullAccess”

Search for “RDS”, check “AmazonRDSFullAccess”

Create group!



Create group

Create a group and select the policies to be attached to the group. Using groups is a b

Group name: administrator

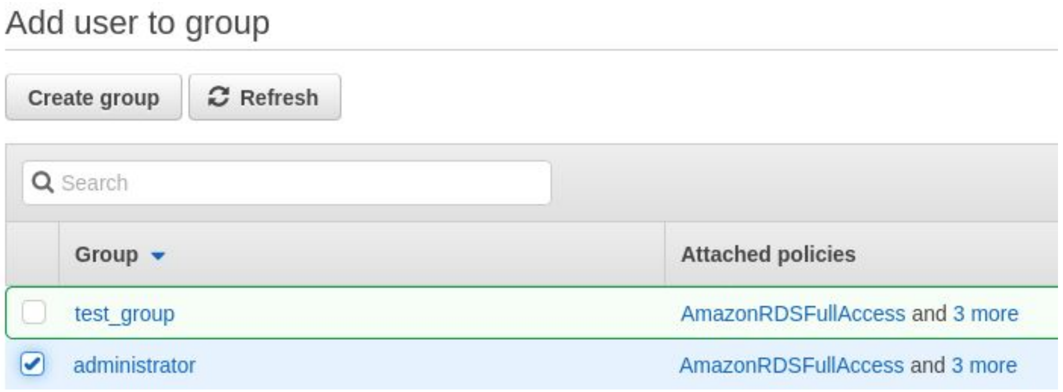
		Policy name	Type
✓	▸	AdministratorAccess	Job function
☐	▸	AdministratorAccess-Amplify	AWS managed
☐	▸	AdministratorAccess-AWSElasticBeanstalk	AWS managed
☐	▸	AlexaForBusinessDeviceSetup	AWS managed
☐	▸	AlexaForBusinessFullAccess	AWS managed
☐	▸	AlexaForBusinessGatewayExecution	AWS managed
☐	▸	AlexaForBusinessLifesizeDelegatedAcce...	AWS managed
☐	▸	AlexaForBusinessPolyDelegatedAccessP...	AWS managed
☐	▸	AlexaForBusinessReadOnlyAccess	AWS managed

Add user to group

The group you created should be selected.

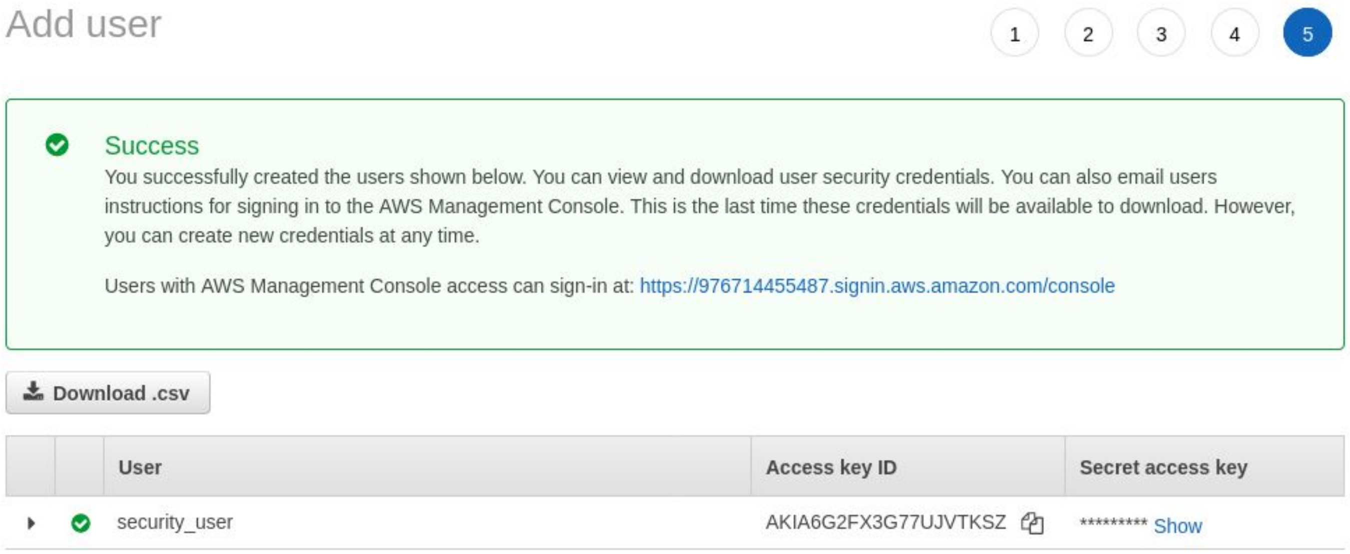
Click Next: Tags

On next page leave blank and click Next: Review



	Group	Attached policies
☐	test_group	AmazonRDSFullAccess and 3 more
✓	administrator	AmazonRDSFullAccess and 3 more

Then create user, save the **Access Key ID** and **Secret access Key**.



Add user

Success

You successfully created the users shown below. You can view and download user security credentials. You can also email users instructions for signing in to the AWS Management Console. This is the last time these credentials will be available to download. However, you can create new credentials at any time.

Users with AWS Management Console access can sign-in at: https://976714455487.signin.aws.amazon.com/console

		User	Access key ID	Secret access key
▸	✓	security_user	AKIA6G2FX3G77UJVTKSZ	********* Show

Connect EB to S3

Open Elastic Beanstalk console

Choose the environment you just created

Go to configuration, under **software**, click Edit

In the **Environment properties** section, define the variables that your application reads to connect to S3.

- **S3_ACCESS_KEY_ID** – The access key ID of the user just created
- **S3_SECRET_KEY** – The secret access key of the user just created
- **S3_BUCKET_NAME** - The name of the bucket created

Apply!

6. Flask code

We have completed settings on AWS. Now, let’s look at the Flask code and make some changes for deployment.

Environment

I recommend you to create a virtual environment for Flask.

We've been running on python 3 even though Werkzeug 0.14.1 only has experimental support for python 3 (https://werkzeug.palletsprojects.com/en/0.14.x/installation/). We have not had any issues.

For example, run *$ virtualenv "name of your environment"*

Activate the environment *$ source ~/<env name>/bin/activate*

Then install all libraries listed in the **requirement.txt** file

Such as *$ pip install Flask;* *$ pip install boto3;* *$ pip install Flask-SSLify*; etc.

Or simply: $ pip install -r requirements.txt

Run the Flask code in this virtual environment

Prepare deploy file

To deploy an application on EB, we need to prepare a .zip file that contains all relevant files and subfolders.

Files:

application.py
Main script, contains all backend logic, database manipulation.

config.py
Configuration file, specify csv settings

We need to make some changes in these two files later.

default_config.py
Configuration file that loads database and S3 connection environment variables

password.npy
Store user login credentials.

data/blobornot_examples.csv
Images and data for all examples to be annotated, the order is the order for annotation, so you should shuffle the list before deployment.

There are four columns:

blobs is the path to 20x images,
10xfields is the path to the 10x field tiles,
coords is the coordinates (x,y,w,h) of the “blobs” image in “10xfields”
rotation is how many degrees the images should be rotated when displayed ie. 0, 90, 180 or 270

Note: Img paths should be the path in the AWS S3 bucket, without bucket name.

requirement.txt
Contains the library names and versions that are used in the application

Subfolders:

database
Contains files for database. The **models.py** defines the tables in the database.

static
Contains static images that are used in the application.

templates
Contains all HTML templates.

data
Contains csvs with blob data needed to display example for annotation

Other scripts

There are three .py files that don’t need to be zipped for deployment.

backup_database.py
Used to backup annotation records into a .csv file. Can be set to run regularly for daily backup.

set_password.py
Set login credentials and unique id for users.

delete_all_annotations.py
Clear the database of all annotations.

zipforaws.sh
Zip all necessary files for Elastic Beanstalk app

Create user credentials

- Edit **set_password.py** with user names and passwords.
- Run `python set_passwords.py` in command line to create passwords.npy

Deploy Flask application locally for testing

You can test the application by running it locally before deploying to the AWS EB environment.

- Set RDS and S3 environment variables locally.
- Set AWS_REGION environment variable locally. See https://docs.aws.amazon.com/general/latest/gr/rande.html for region codes
- To start flask application locally, use the following command in the same directory as application.py

```
BLOBORNOT_RUN_LOCAL=TRUE flask run
```

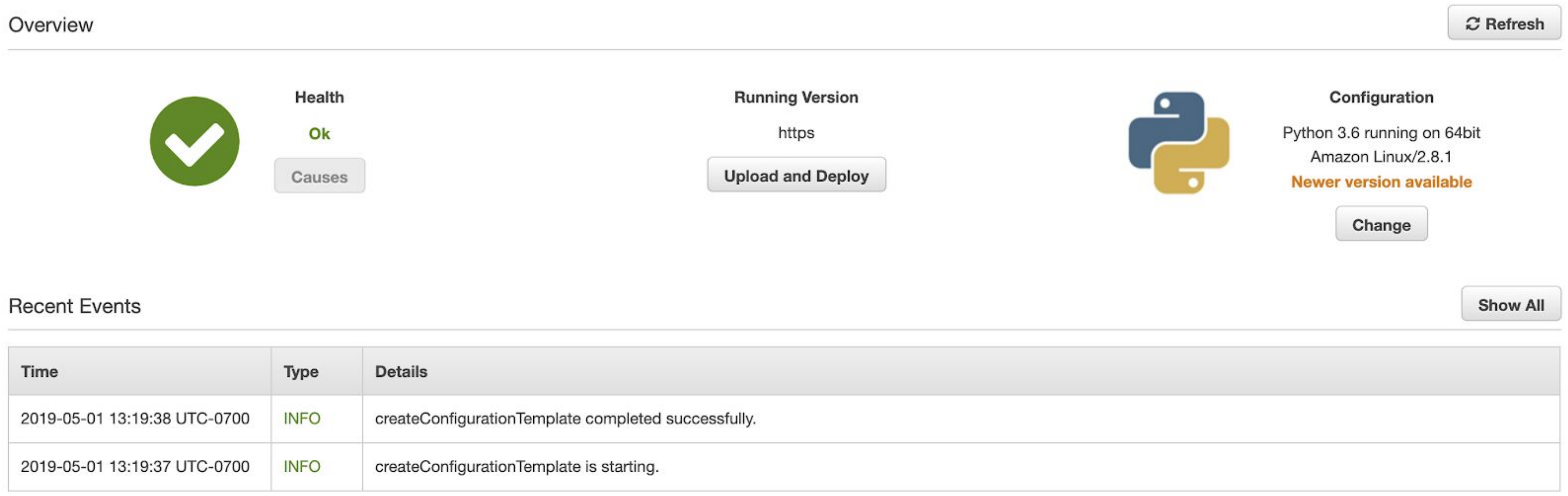
Deploy Flask application on EB

Zip all relevant files
on linux and mac, you can run zipforaws.sh from within same directory as application.py

Go to the EB dashboard

Click **upload and deploy**

Upload the .zip file you created



Python 3.6 running on 64bit Amazon Linux/2.8.1

Time	Type	Details
2019-05-01 13:19:38 UTC-0700	INFO	createConfigurationTemplate completed successfully.
2019-05-01 13:19:37 UTC-0700	INFO	createConfigurationTemplate is starting.

7. HTTPS

If the application can run correctly, we can now allow HTTPS for it.

Register a domain

To allow HTTPS, you need to have a permanent domain. I’ve already registered a domain “blobornot.com” for the project, but the control is under my AWS account. We should be able to transfer this domain to a new AWS account

If you’d like to register a new domain,

Go to Route 53 https://console.aws.amazon.com/route53/

Under **Domain Registration**, choose **Get Started Now**.

Choose **Register Domain**

Set the domain name you’d like to use

Fill out contact details, done!

Request a public certificate

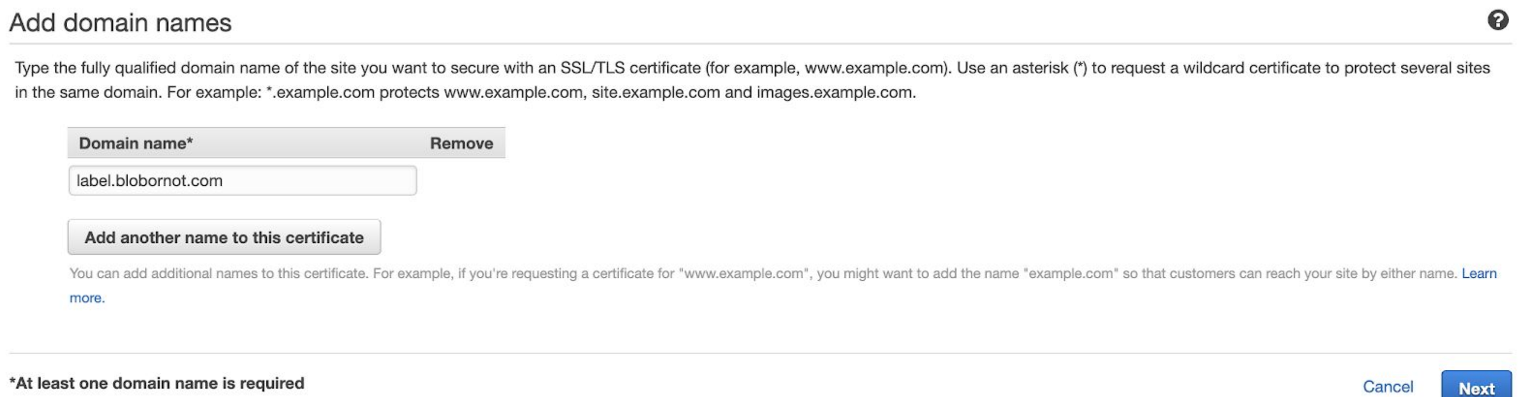
Go to ACM https://console.aws.amazon.com/acm/home

If prompted, under Provision certificates, click Get Started

Click **request a certificate**

Add domain name “XXX.blobornot.com” (or replace with your own domain)

I’m using “label.blobornot.com” for phase1



Add domain names

Type the fully qualified domain name of the site you want to secure with an SSL/TLS certificate (for example, www.example.com). Use an asterisk (*) to request a wildcard certificate to protect several sites in the same domain. For example: *.example.com protects www.example.com, site.example.com and images.example.com.

Domain name*	Remove
label.blobornot.com	

You can add additional names to this certificate. For example, if you're requesting a certificate for "www.example.com", you might want to add the name "example.com" so that customers can reach your site by either name. Learn more.

*At least one domain name is required

Request a public certificate

If you have already registered a new domain, you can choose either “DNS validation” or “Email validation” for certificate verification.

Allow load balancer

Open Elastic Beanstalk console

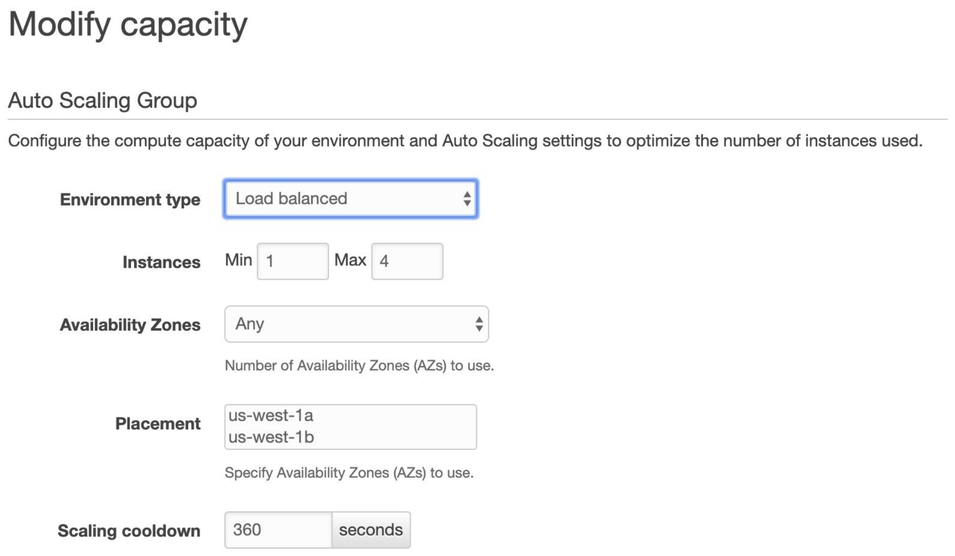
Go to configuration

Under **capacity**, click modify

Change the environment type to “**load balanced**”

Apply!

Confirm!



Modify capacity

Auto Scaling Group

Configure the compute capacity of your environment and Auto Scaling settings to optimize the number of instances used.

Environment type	Load balanced
Instances	Min 1 Max 4
Availability Zones	Any Number of Availability Zones (AZs) to use.
Placement	us-west-1a us-west-1b Specify Availability Zones (AZs) to use.
Scaling cooldown	360 seconds

Add Load balancer

Go back to EB configuration

Under load balancer, click modify

Under **Classic Load Balancer**

a. Choose **Add listener**.
b. In the **Classic Load Balancer listener** dialog box, configure the following settings:
   - For **Listener port**, type the incoming traffic port, typically `443`.
   - For **Listener protocol**, choose **HTTPS**.
   - For **Instance port**, type `80`.
   - For **Instance protocol**, choose **HTTP**.
   - For **SSL certificate**, choose the certificate you just created.
   - Under **Sessions** select Stickiness policy enabled
c. Choose **Add**.

Apply!

Alias to EB environment

Go to Route 53 https://console.aws.amazon.com/route53/

Click **hosted Zones**, click your domain

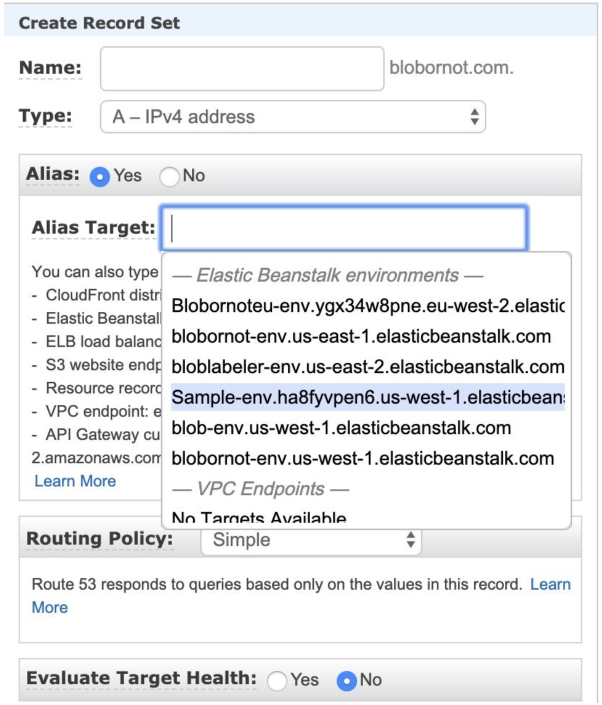
Click **create record set**

On the right, type your domain name for public certificate

Check **Yes** for **Alias**

Select the EB environment you created

Create!



Create Record Set

Name: blobornot.com.

Type: A – IPv4 address

Alias: [✓ Yes] No

Alias Target:

You can also type
- CloudFront distri
- Elastic Beanstal
- ELB load balanc
- S3 website endp
- Resource record
- VPC endpoint: e
- API Gateway cu

2.amazonaws.com

Learn More

— Elastic Beanstalk environments —

Blobornoteu-env.ygx34w8pne.eu-west-2.elastic

blobornot-env.us-east-1.elasticbeanstalk.com

bloblabeler-env.us-east-2.elasticbeanstalk.com

Sample-env.ha8fyvpen6.us-west-1.elasticbean

blob-env.us-west-1.elasticbeanstalk.com

blobornot-env.us-west-1.elasticbeanstalk.com

— VPC Endpoints —

No Targets Available

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. Learn More

Evaluate Target Health: Yes [✓ No]

The domain name (“XXX.blobornot.com” or other name) you set will now alias to the EB application

Now you should be able to access the application through the new url “XXX.blobornot.com” and it will support HTTPS.

Backup annotations to csv

Use backup_database.py to backup annotations in the database to a csv on a local machine

- Set RDS and S3 environment variables locally.
- Set AWS_REGION environment variable locally. See https://docs.aws.amazon.com/general/latest/gr/rande.html for region codes
- Run the following command in the same directory as backup_database.py

```
python backup_database.py
```

Annotations are stored in a file in the same directory. Edit backup_database.py to change this directory and file name.