



# Certified Tech Developer

The Ultimate Degree

## Infraestructura II

# Ejercitación GitLab + AWS S3

### OBJETIVO

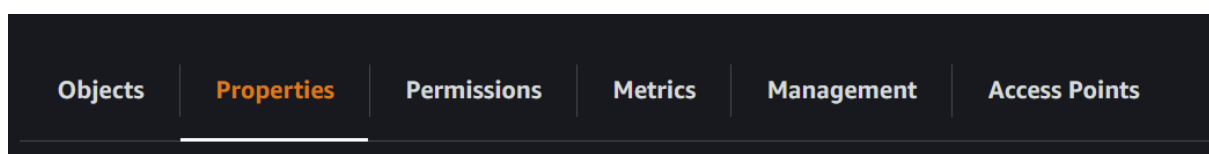
Generar un pipeline en GitLab que me permita pushear el repositorio a un bucket s3 configurando el bucket como *static website hosting* para poder visualizar la página públicamente.

### RESOLUCIÓN

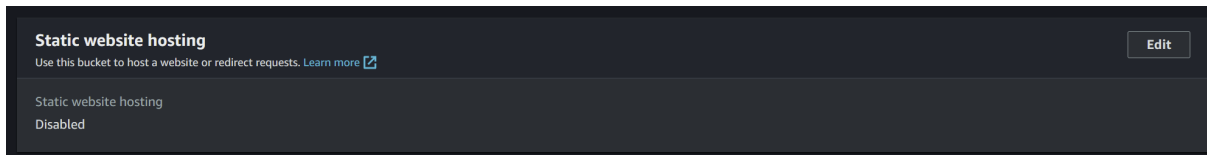
Continuamos integrando herramientas. En nuestro ambiente de GitLab debemos subir a un nuevo proyecto un template web. Podemos obtenerlo desde esta url: <https://plantillashtmlgratis.com/en/home/>. Tendremos que configurar un nuevo runner.

Antes de realizar todo esto, debemos crear nuestro bucket S3 en AWS y setearlo para hospedar páginas web estáticas.

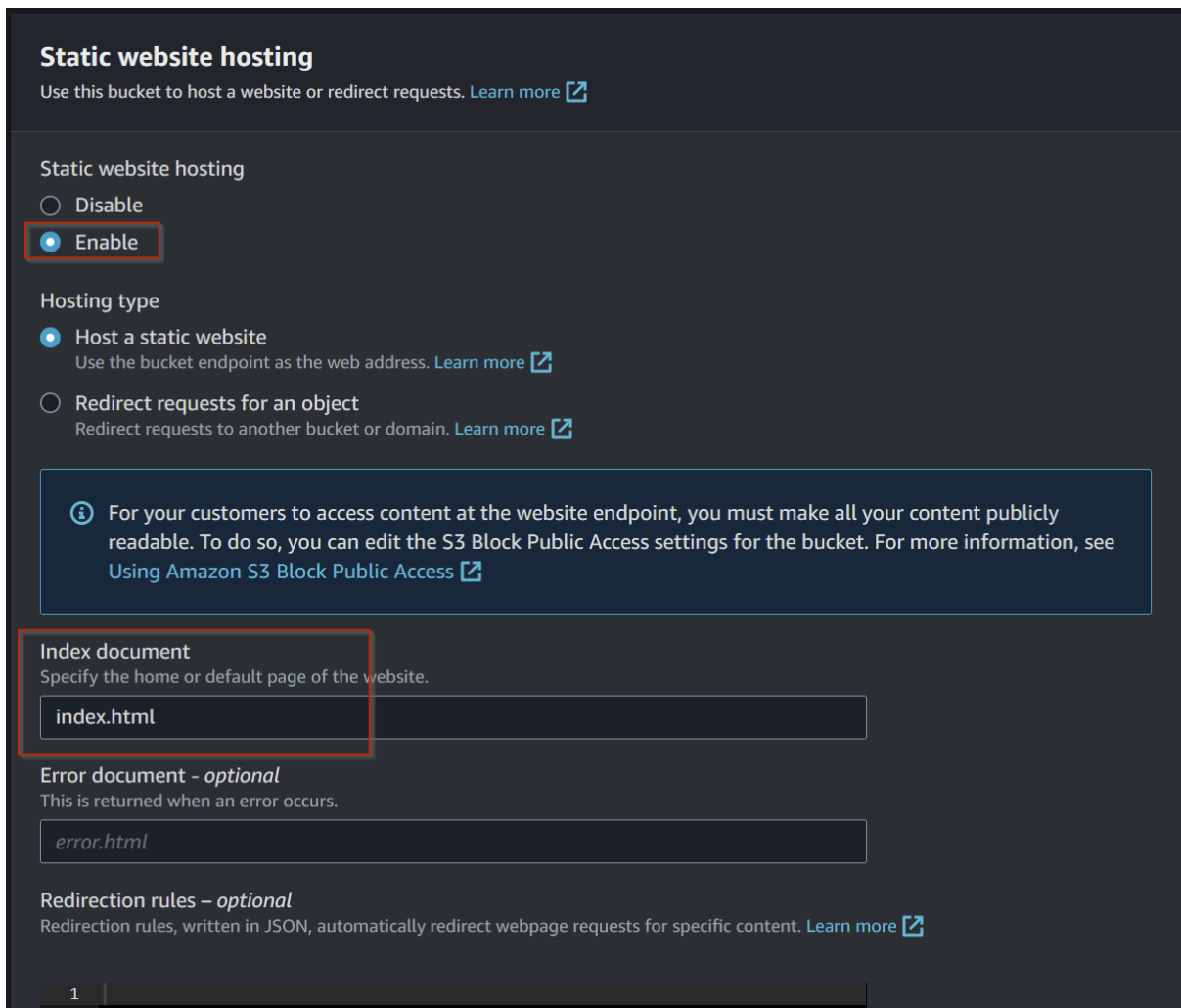
Esto lo hacemos desde la solapa propiedades del bucket



Y vamos hasta el final hasta encontrar la siguiente opción:



Una vez que habilitamos la opción, indicamos en la documentación del index el archivo principal de nuestra web que sería el index.html como se muestra en la imagen siguiente:



Verificamos que nuestro bucket posea acceso al público.

## Edit access control list (ACL) [Info](#)

### Access control list (ACL)

Grant basic read/write permissions to other AWS accounts. [Learn more](#)

Grantee	Objects	Bucket ACL
<b>Bucket owner (your AWS account)</b> Canonical ID:  7abddac41e55ea7fda3111d64c07e9db88d2c37e623524c0b1e37b16ab5b0f8b	<input checked="" type="checkbox"/> List <input checked="" type="checkbox"/> Write	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write
<b>Everyone (public access)</b> Group:  http://acs.amazon.com/groups/global/AllUsers	<input checked="" type="checkbox"/> List <input type="checkbox"/> Write	<input type="checkbox"/> Read <input type="checkbox"/> Write

Y que no tenga bloqueos:

**Block all public access**  
 Off

▼ Individual Block Public Access settings for this bucket

- ☐ 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.

Generamos nuestra policy para el bucket del generador de policy de aws

**amazon web services**

## AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see key concepts in Using AWS Identity and Access Management. Here are sample policies.

### Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy: **S3 Bucket Policy**

### Step 2: Add Statement(s)

A statement is the formal description of a single permission. Statements are added to a policy to define the permissions that the policy grants.

Effect: ☒ Allow ☐ Deny

Principal: \*

AWS Service: **Amazon S3**

Actions: **1 Action(s) Selected**

Amazon Resource Name (ARN): **arn:aws:s3:::gitlab-bucket**

Add Conditions (Optional):

**Policy JSON Document**

```
{
  "Id": "Policy1679335287350",
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1679335287350",
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::gitlab-bucket-mcatrina/*",
      "Principal": "*"
    }
  ]
}
```

You added the following statements. Click the button below to generate the policy.

Principal(s)	Effect	Actions	Conditions
*	Allow	s3:GetObject	None

### Step 3: Generate Policy

A policy is a document (written in the [Access Policy Language](#)) that acts as a container for one or more statements.

**Generate Policy** **Start Over**

This AWS Policy Generator is provided for informational purposes only, you are still responsible for your use of Amazon Web Services technologies and ensuring that your use is in compliance with all applicable terms and conditions. This AWS Policy Generator is provided as is without warranty of any kind, whether express, implied, or statutory. This AWS Policy Generator does not modify the applicable terms and conditions governing your use of Amazon Web Services technologies.

©2010 Amazon Web Services LLC or its affiliates. All rights reserved.

Y nos quedaría de la siguiente manera:

### Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

**Edit** **Delete**

```
{
  "Version": "2012-10-17",
  "Id": "Policy1679335290034",
  "Statement": [
    {
      "Sid": "Stmt1679335287350",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::gitlab-bucket-mcatrina/*"
    }
  ]
}
```

**Copy**

Una vez ya teniendo nuestro bucket configurado procedemos a levantar un runner y configurarlo.

```
docker run -d --name gitlab-runner \
-v /srv/gitlab-runner/config:/etc/gitlab-runner \
-v /var/run/docker.sock:/var/run/docker.sock \
gitlab/gitlab-runner:latest
```

Ingresamos al contenedor y registramos el Runner pero configuramos el executor como **docker** y default image **python:3.6**

Sino recordamos con que comando registrar el runner es el siguientes:

```
gitlab-runner register
```

Luego generamos nuestro archivo `.gitlab-ci.yml` dentro de nuestro repositorio de gitlab con los siguientes datos:

**stages:**

- despliegue

**deploy:**

**stage:** despliegue

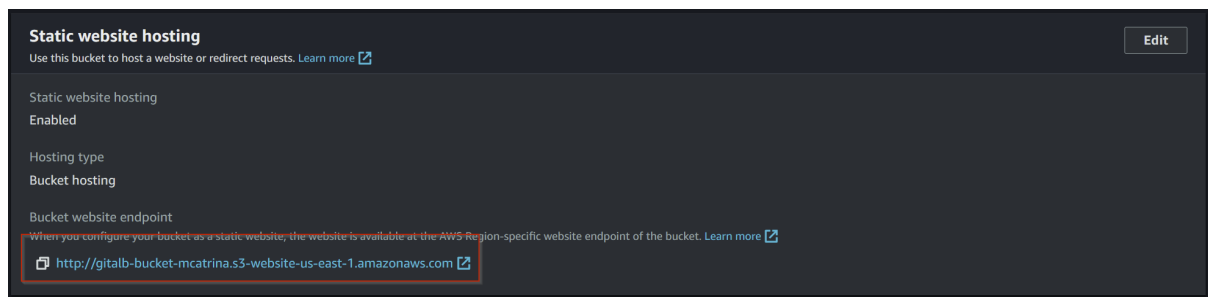
**image:** python:3.6

**script:**

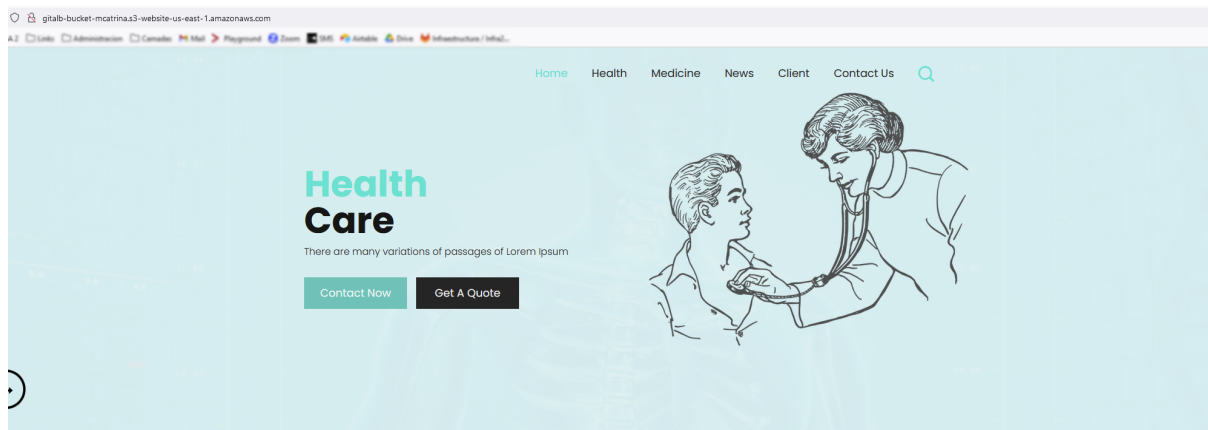
- echo "Installing aws command line"
- pip install awscli
- aws --version
- echo "Extracting the production build"
- echo "Uploading files to AWS Bucket"
- aws s3 cp --recursive . s3://<nuestro\_bucket>

Generar las variables de AWS en la solapa de Variables en GITLAB.

Una vez ya nuestro pipeline se ejecutará de manera automática y para poder visualizar nuestra web desde el bucket tenemos que buscar el link desde el siguiente lugar:



Y ya tenemos nuestra web funcionando:



## Best Of Health care for you

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis



## AYUDA

No nos olvidemos de que tenemos que configurar la red de nuestro runner y del contenedor. Dentro del contenedor del runner modificamos el archivo `/etc/gitlab-runner/config.toml` agregar el `network_mode = <nuestra red creada del contenedor>` al final de la línea.

```
[session_server]
  session_timeout = 1800

[[runners]]
  name = "26b7c7e47eb1"
  url = "http://gitlab"
  id = 2
  token = "aragre6crcBXjqjTB5cF"
  token_obtained_at = 2023-01-10T06:29:53Z
  token_expires_at = 0001-01-01T00:00:00Z
  executor = "docker"
  clone_url = "http://gitlab"
  [runners.custom_build_dir]
  [runners.cache]
    MaxUploadedArchiveSize = 0
  [runners.cache.s3]
  [runners.cache.gcs]
  [runners.cache.azure]
  [runners.docker]
    tls_verify = false
    image = "ruby:2.7"
    privileged = false
    disable_entrypoint_overwrite = false
    oom_kill_disable = false
    disable_cache = false
    volumes = ["/cache"]
    shm_size = 0
    network_mode = "gitlab-network"
```

Tanto el contenedor de Gitlab y el del Runner, tienen que estar dentro de la misma red.

Crear la red con **"docker network create <nombre>"**

Conectarlo con **"docker network connect <nombre de la red creada>  
<contenedor id>"**

Pueden verificarlo con el comando **"docker network inspect <contenedor id>"**

---

Las políticas que utilice dentro del bucket S3 fueron la siguientes:

```
{  
  "Version": "2012-10-17",  
  "Id": "Policy1679335290034",  
  "Statement": [  
    {  
      "Sid": "Stmt1679335287350",  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": "s3:GetObject",  
      "Resource": "arn:aws:s3:::gitalb-bucket-mcatrina/*"  
    }  
  ]  
}
```

Les tocaría cambiar en la parte del Resource el nombre del bucket que ustedes tienen. En caso de que quieran averiguar de donde se obtiene, es desde las propiedades del bucket como lo muestro en la siguiente imagen:

Objects	Properties	Permissions	Metrics	Management	Access Points
Bucket overview					
AWS Region		Amazon Resource Name (ARN)		Creation date	
US East (N. Virginia) us-east-1		arn:aws:s3:::gitalb-bucket-mcatrina		March 11, 2023, 01:34:03 (UTC-03:00)	