Reporte de práctica de laboratorio

Práctica 10: Control de versiones (CodeCommit)

Fecha 01/04/2021

Maestría en Informática Aplicada

*Aplicaciones y Servicios en la Nube*

Prof. Mtro. Rodolfo Luthe Ríos

Roberto Leos

[mi41587@iteso.mx](mailto:mi41587@iteso.mx)

# Introducción

En esta practica vamos a trabajar con el Git local el cual nos va a ayudar a tener control de versiones de los archivos que subamos al folder destinado. Después haremos el mismo proceso, pero usando el GitHub el cual puedes configurar el repositorio en la nube de ellos y hacerlo públicos o privados. Después haremos la conexión al AWS CodeCommit y en todos debemos ser capaces de ver los cambios reflejados en los archivos.

Objetivos

• Utilizar un cliente de control de versiones

• Configurar servicios de control de versiones centralizados

• Controlar las versiones de un documento

Actividades

• Configurar un repositorio local

• Crear un repositorio en GitHub

• Crear un repositorio en CodeCommit

• Controlar las versiones de un documento

# Marco Teórico

De acuerdo con Prashant Agrawal [1], La competencia de la organización para brindar servicios y aplicaciones a alta velocidad requiere competir de manera efectiva en el mercado. Las prácticas y herramientas para tales procesos de gestión exigen un modelo rápido y confiable. Los cambios deben comenzar en el nivel de ingeniería de software al crear aplicaciones en la nube, por lo que es necesario automatizar nuestros procesos de DevOps utilizando herramientas de automatización de DevOps en la nube y fuera de la nube.

DevOps, un sistema de pensamiento con una preocupación principal por desarrollar, implementar y operar software de alta calidad. Si considera el desarrollo, la implementación y las operaciones como una canalización para que su código fluya, DevOps se trata de mirar esa canalización desde una perspectiva holística. El objetivo de considerar la tubería de manera integral es encontrar formas de hacerla más eficiente y producir productos de mayor calidad.

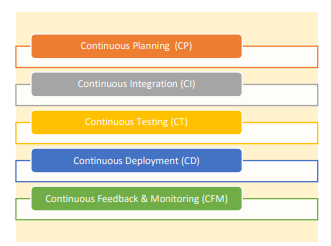


Figura 1. DevOps framework, Prashant Agrawal [1]

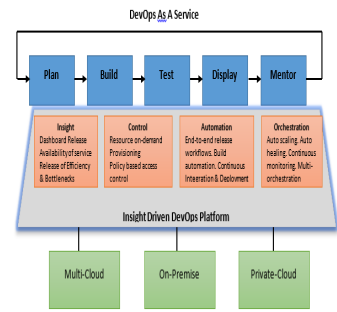
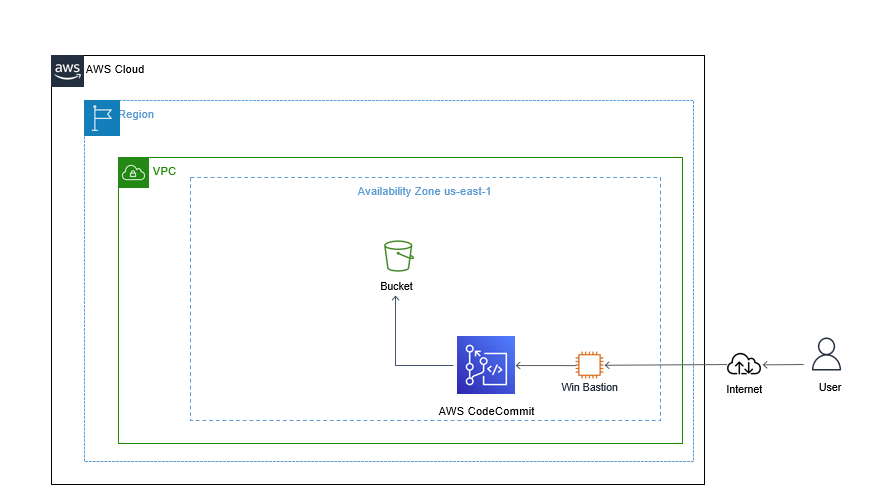


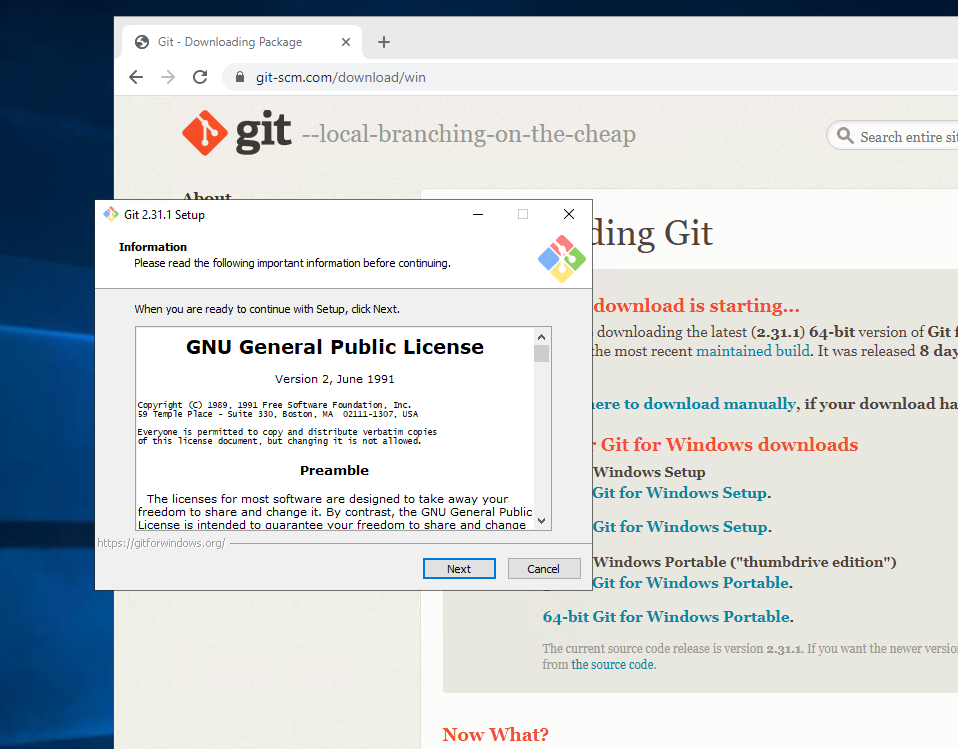
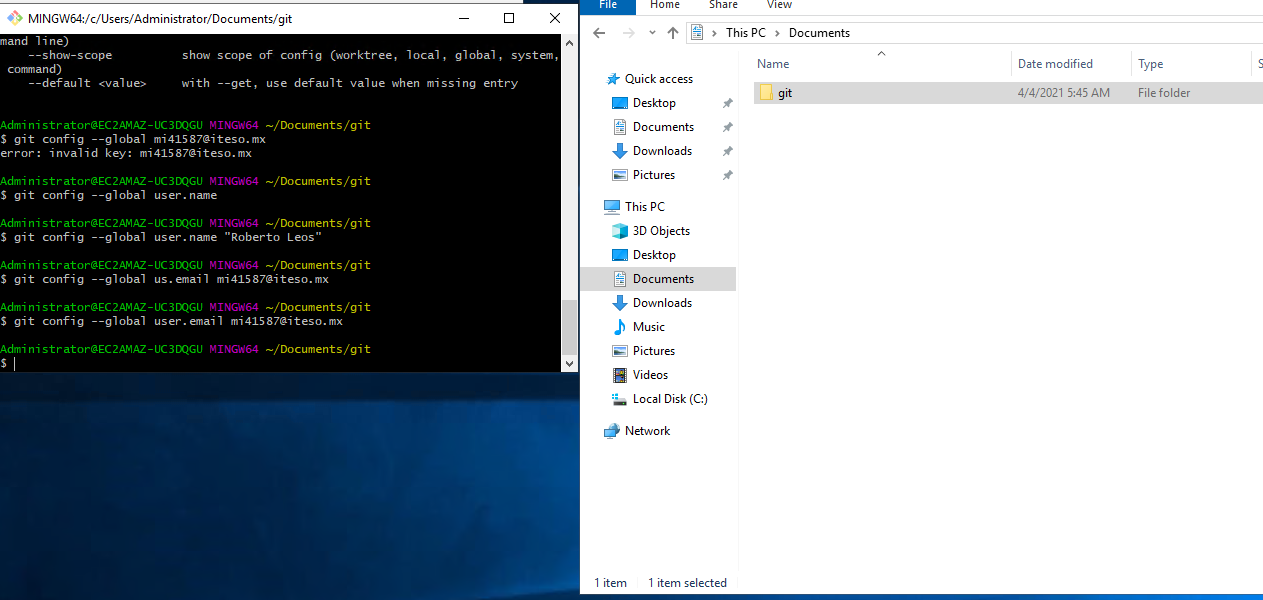
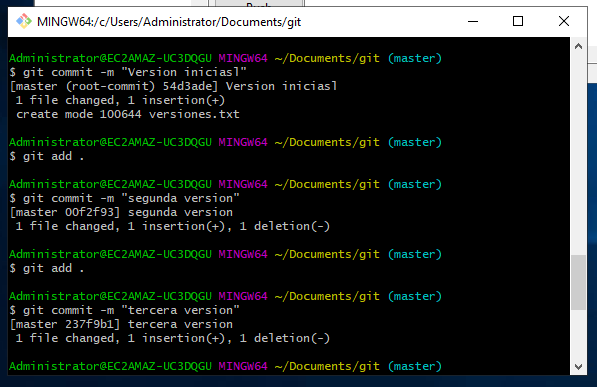
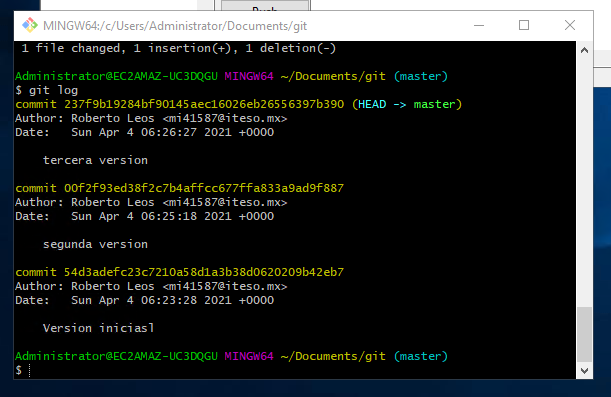
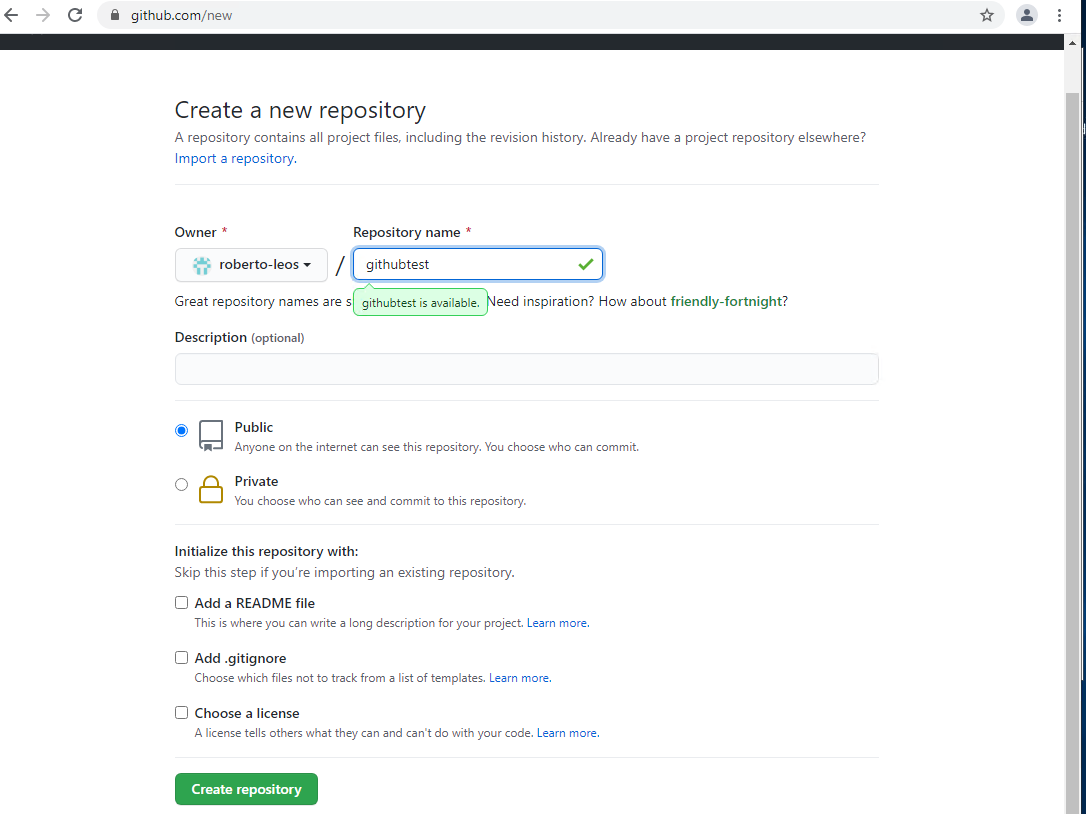
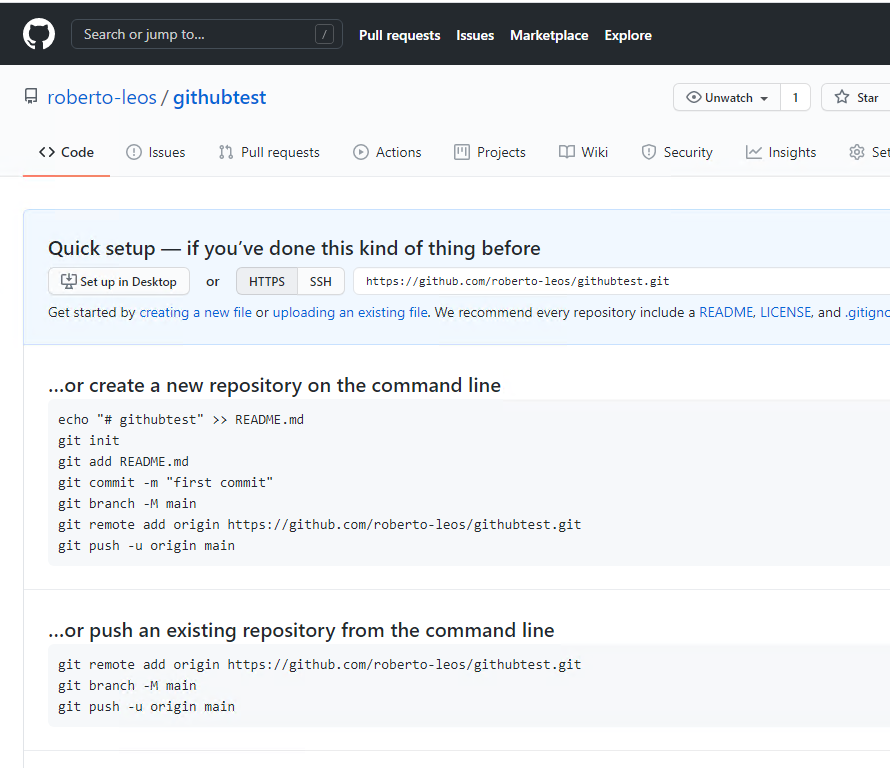
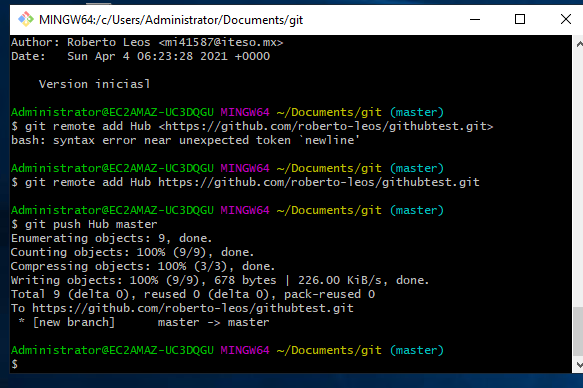
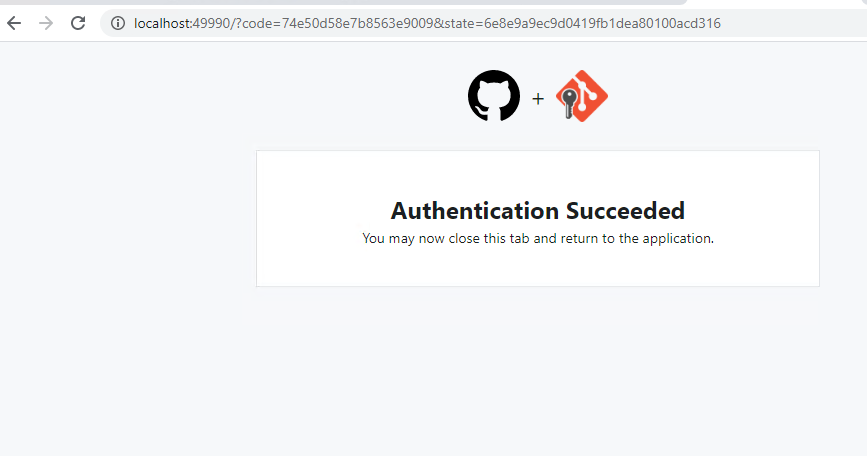
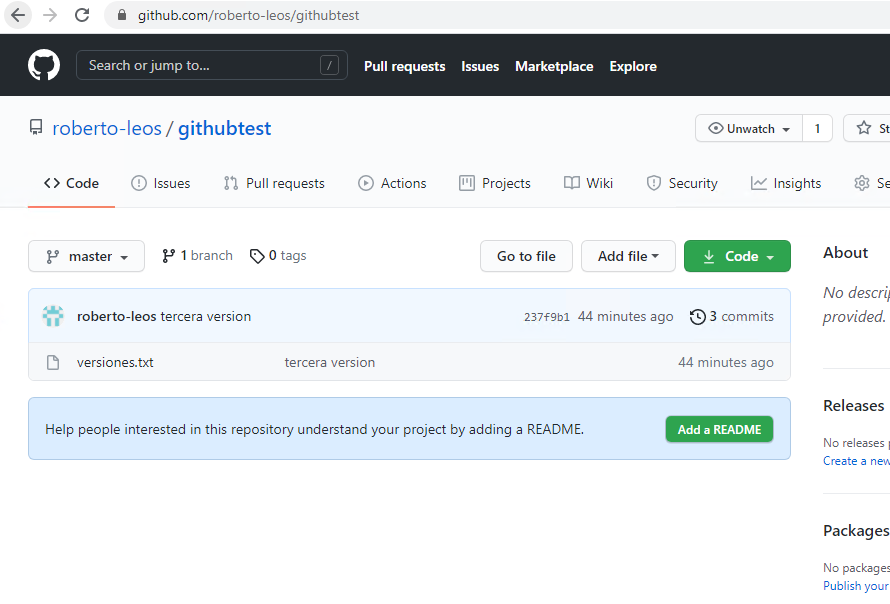
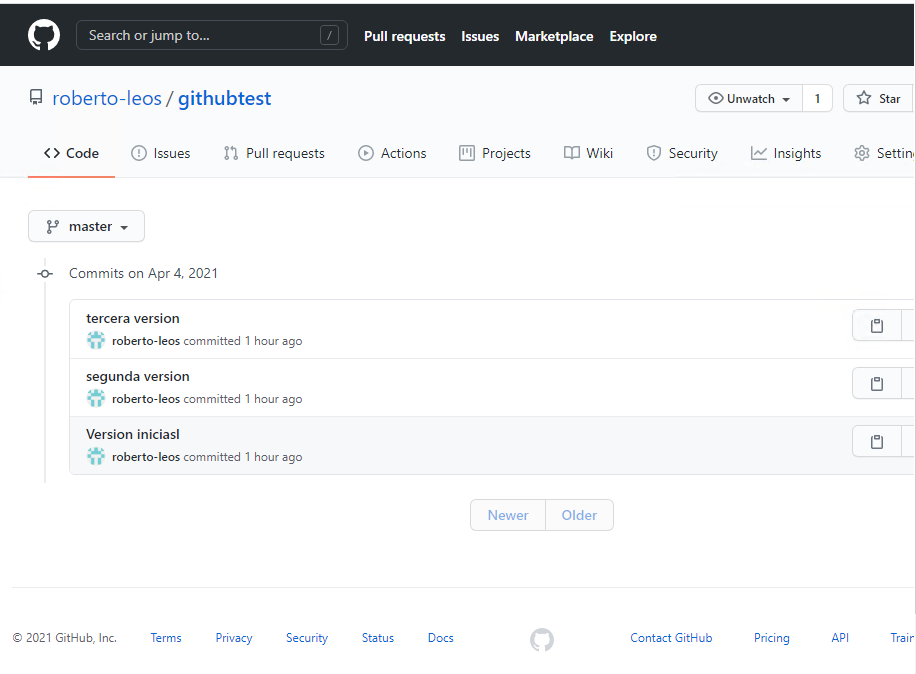
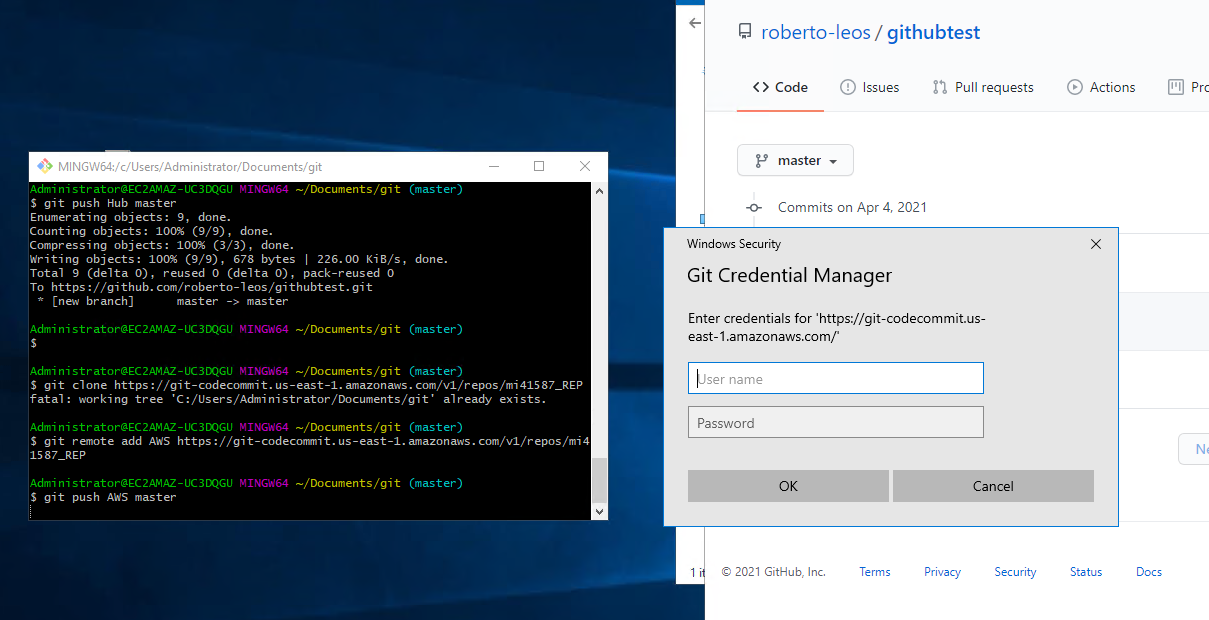
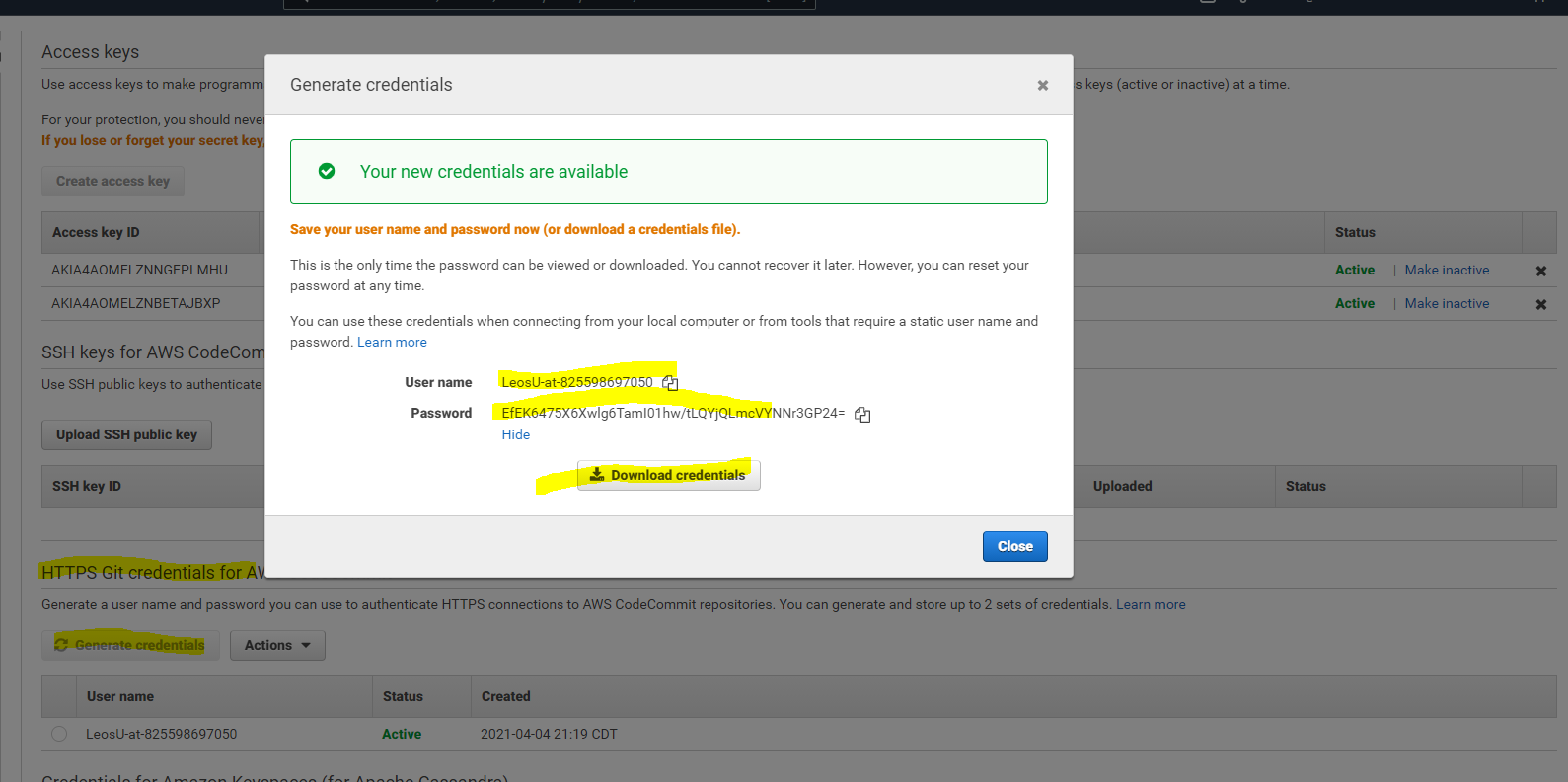
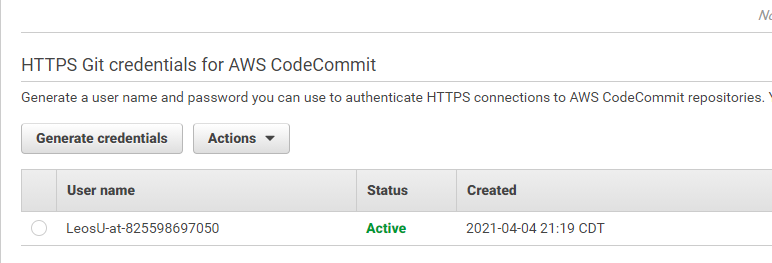
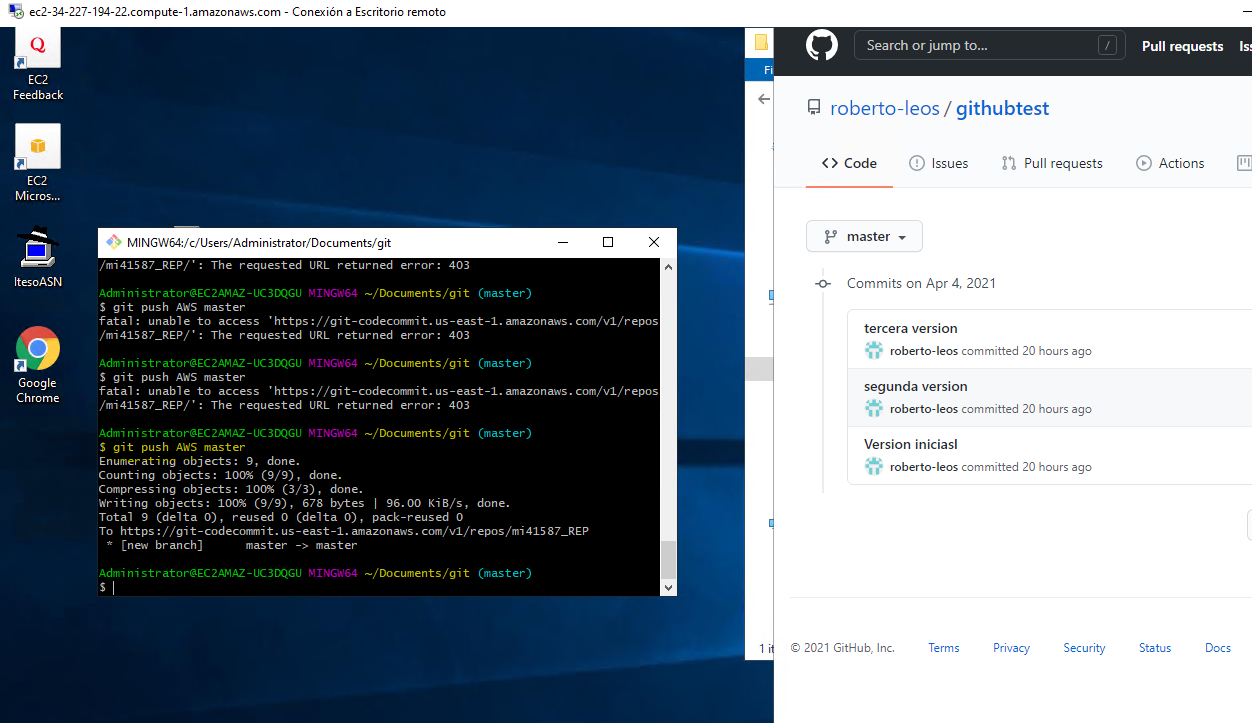
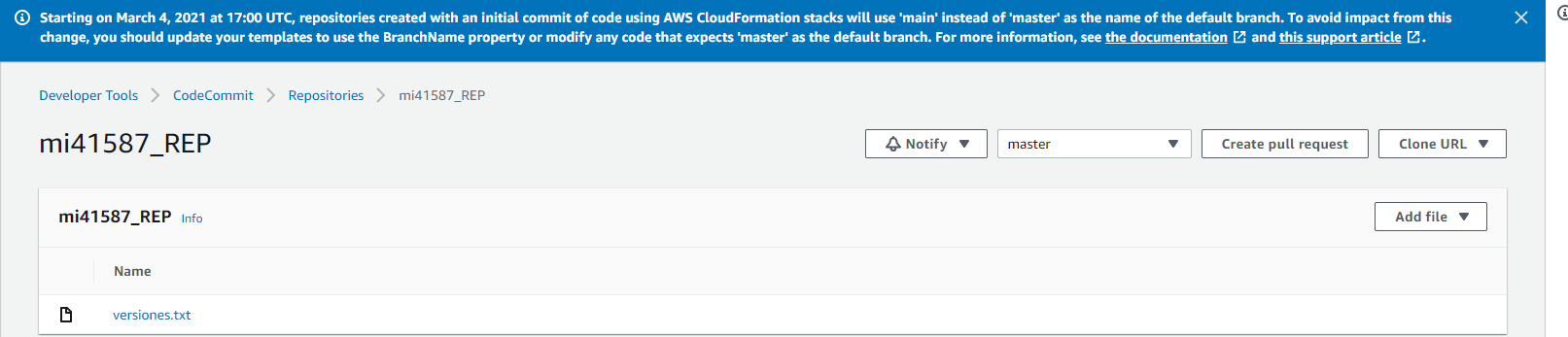
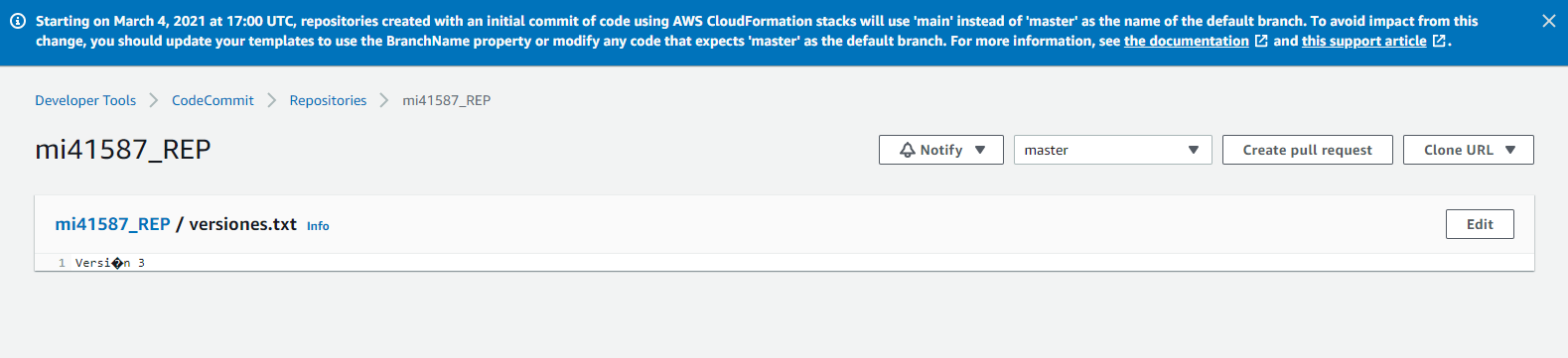
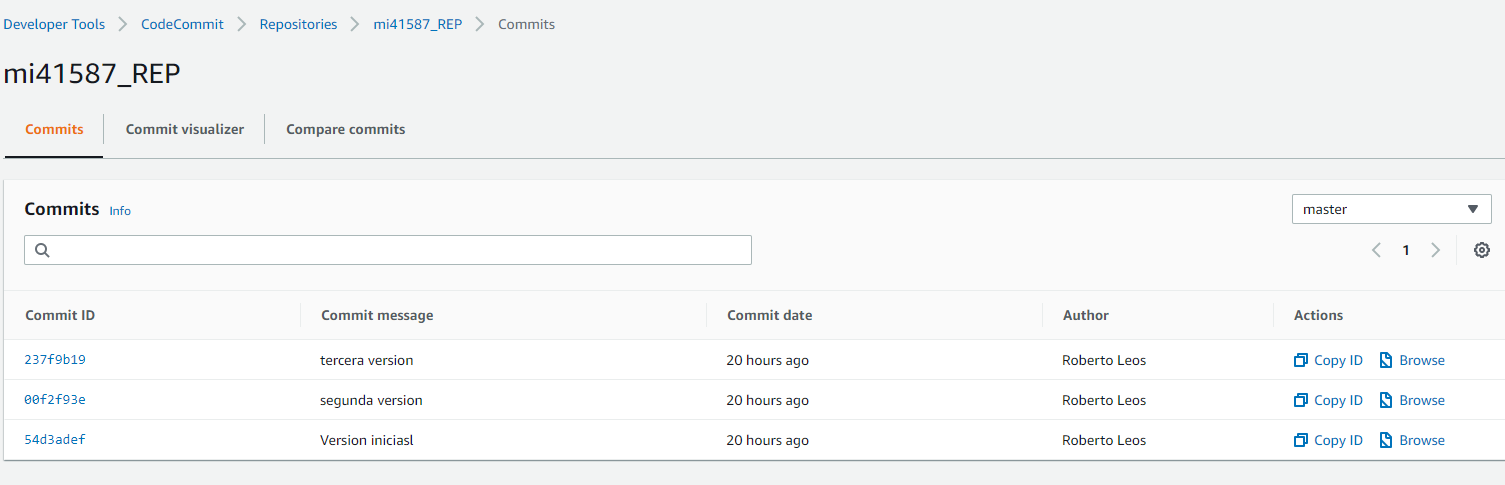
Figura 2, DevOps as a services. Prashant Agrawal [1]

Adoptar metodologías DevOps para proyectos piloto con Cloud Computing demuestra ser un gran éxito en términos de las siguientes formas: Aceleración de la innovación, los ingenieros de DevOps pueden captar fácilmente el impacto de los cambios en las aplicaciones. Los reparadores de software son comparativamente más rápidos. Mejore la colaboración, aumente la eficiencia, reduzca las fallas: el enfoque DevOps promueve lanzamientos de código más frecuentes, ya que tanto los ciclos de desarrollo como los de operación están asociados con este enfoque en la plataforma en la nube. Acelere el tiempo de recuperación: la implementación de DevOps generalmente está aislada y dirigida, por lo que los errores son más fáciles de detectar y, a su vez, más fáciles de corregir e implementar. Mejorar la satisfacción laboral: DevOps promueve el entorno de bases de rendimiento, lo que reduce los obstáculos burocráticos y fomenta el intercambio de riesgos.

# Diagrama

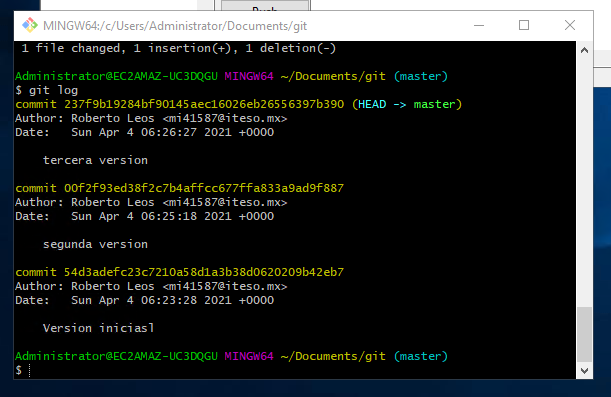


# Desarrollo de la Práctica.

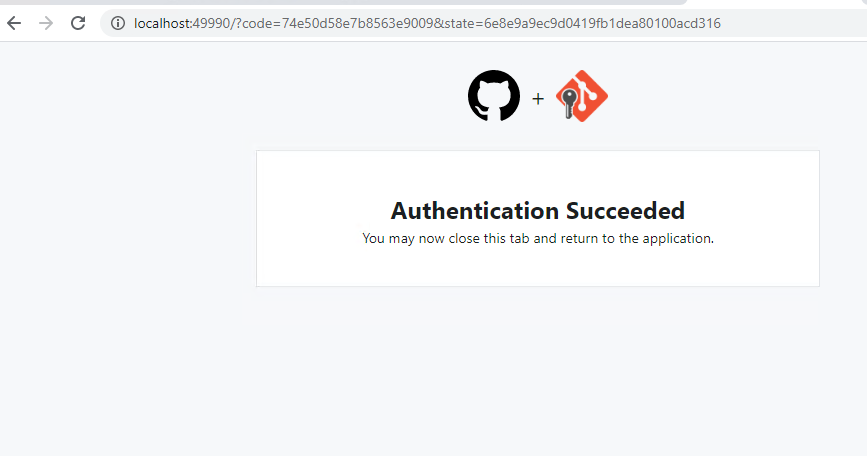
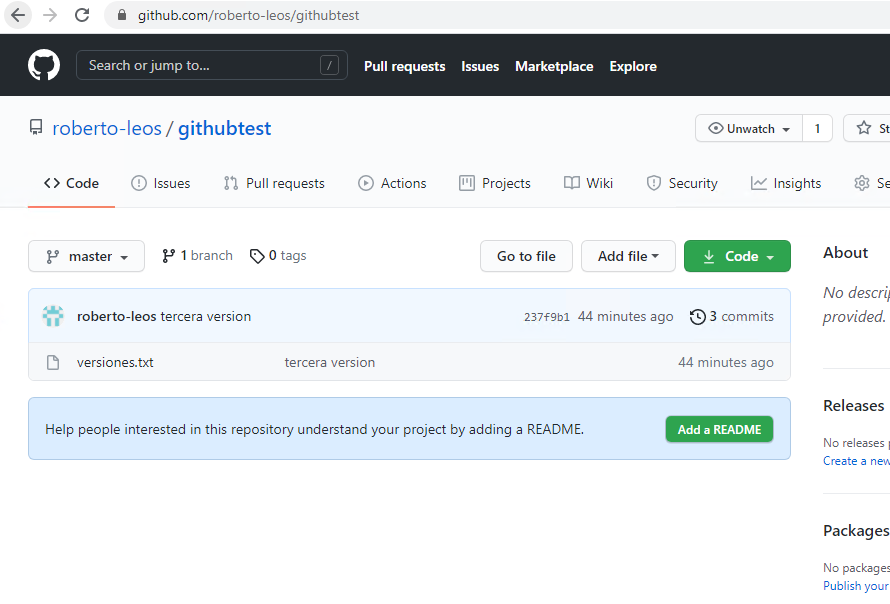
1. Instalar cliente [git](https://git-scm.com)  
   
2. Vamos a configurar el git con cuenta del iteso y vamos a crear un folder nuevo en mis documentos llamada git a la cual dar botón derecho en el folder y seleccionar “git bash here”.comando “git config --global user.name "Roberto Leos" y git config --global user.email mi41587@iteso.mx”  
   
3. Se va crear un archivo .txt en el directorio git llamado versiones.txt con el contenido “versión 1”, luego daremos seguimiento al archivo desde el git dando el comando “git add .” desde la consola y luego poner un comentario “git commit -m “versión inicial”” vamos a repetir este paso en tres ocasiones cambiando el contenido del archivo “versiones.txt” y después consultaremos la historia con el comando “git log”  
     
   
4. Ahora crearemos una cuenta en GitHub  
   
5. Crearemos un repositorio  
   
6. Con el url que da el repositorio la vamos a ligar desde nuestra carpeta git usando el comando “git remote add Hub <https://github.com/roberto-leos/githubtest>” y luego “git push Hub master” con este va pedir dar permisos para hacer el link y veremos nuestro archivo “versiones.txt2 en el Github”  
     
     
   
7. Podemos ver el archivo con todos y la historia de los tres cambios que se hicieron.   
   
8. 
9. Siguiendo la guía “[setup for HTTPSUsers using Git credentials](https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html?icmpid=docs_acc_console_connect_np)” en al IAM después de agregar la política “AWSCodeCommitPowerUser” a tu usuario vamos a ir al tap “security credentials” y en la sección “HTTPS Git credentials for AWS CodeCommit” le damos al botón “Generate credentials”. Te va generar el usuario y password y lo podrás descargar en .csv   
     
   
10. Una vez generado el acceso podemos dar el “git remote add AWS <URL>” y después el “git push AWS master” el cual te pedirá las credenciales antes mencionadas y podrás ver el archivo “versiones.txt” en nuestro repositorio en AWS.   
      
      
      
    
11. Versiones del entregable de la practica 10 en donde por cada sección del documento vamos hacer add y commit para mostrar el log de cambios.
12. ¿Ves conveniente el uso de control de versiones para documentos que no son código?
    1. Si, por ejemplo, en documentación de proyectos, en áreas como control de documentos para los procedimientos. En general esto es muy útil para llevar una correcta documentación sobre cualquier cosa.
13. ¿Cómo comparas GitHub y CodeCommit? Los dos son implementaciones DVCS de git
    1. Los dos te dejan hacer cargas en la nube y compartir los archivos desde un repositorio, pero veo a codecommit mas con el proceso devops pues maneja AWS CodeStar, CodePipeline, CodeCommit, CodeBuild, CodeDeploy y en github solo encontré opciones de Kanban en los templates.

# Experimentos y Resultados.

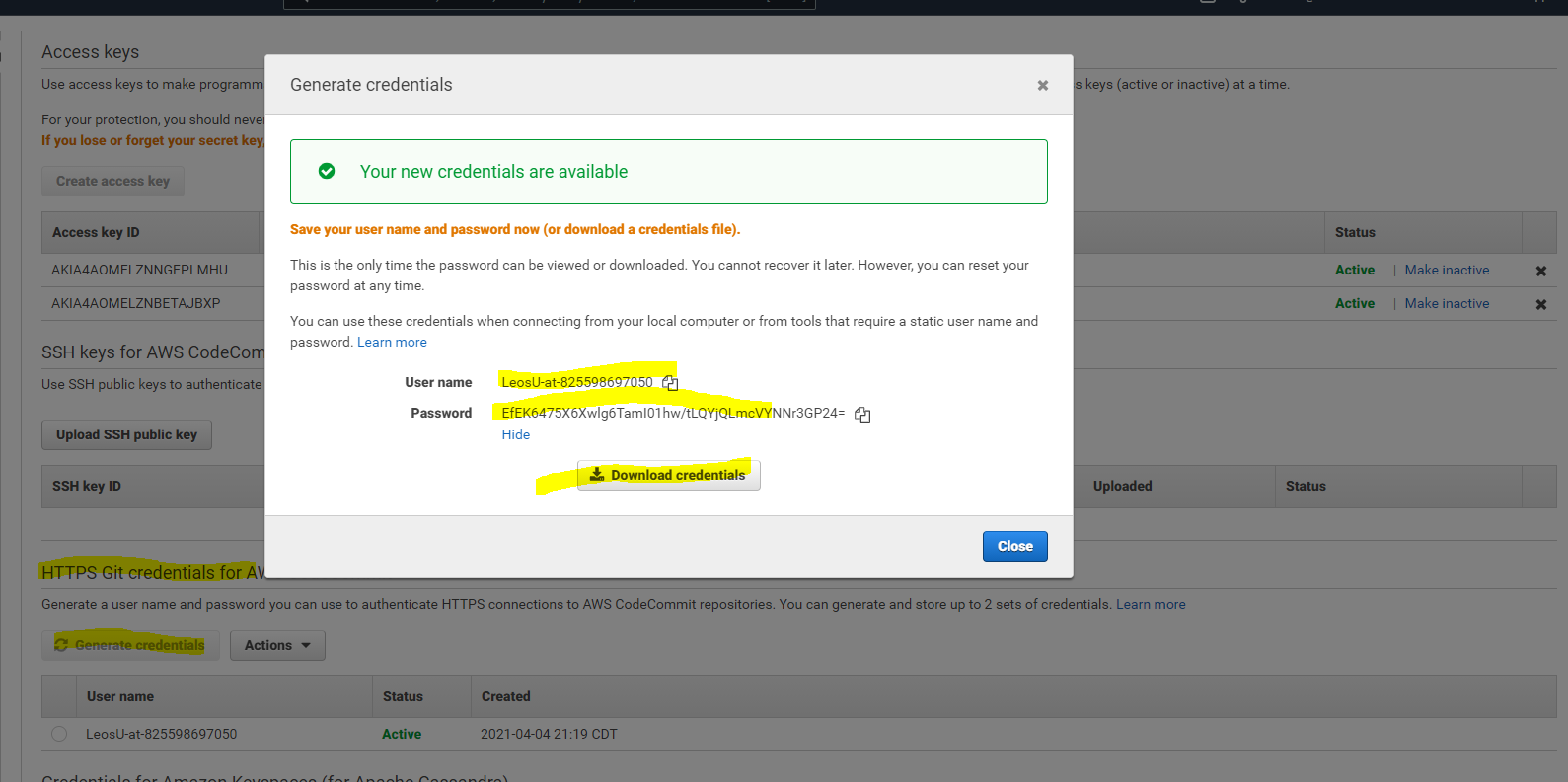
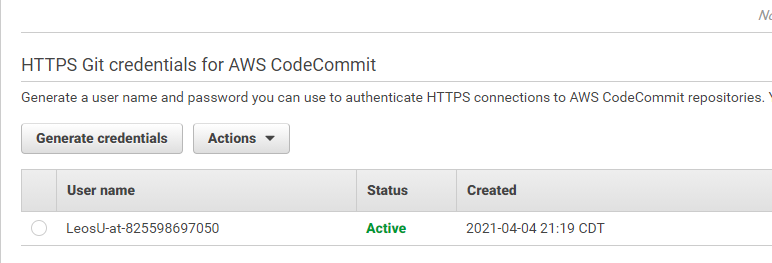
Git Log

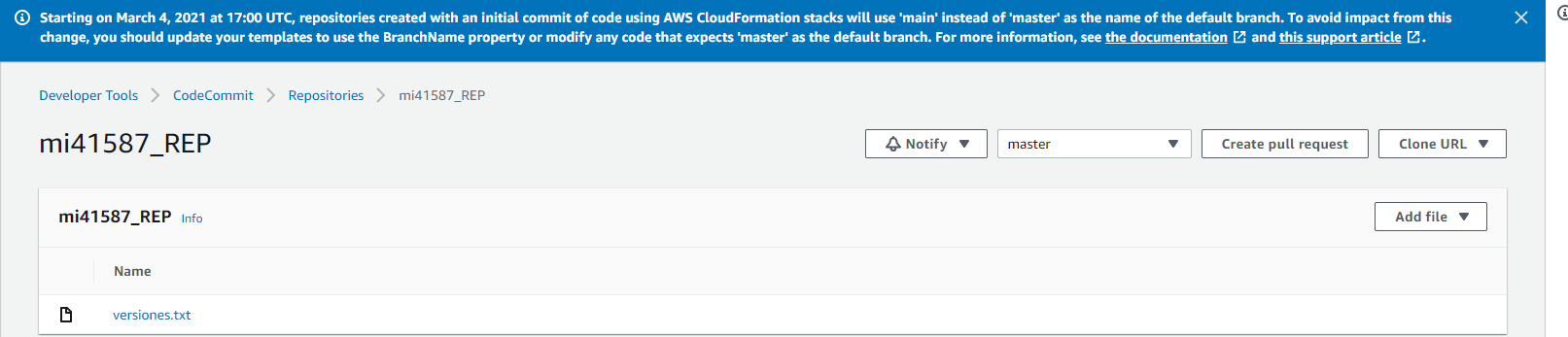
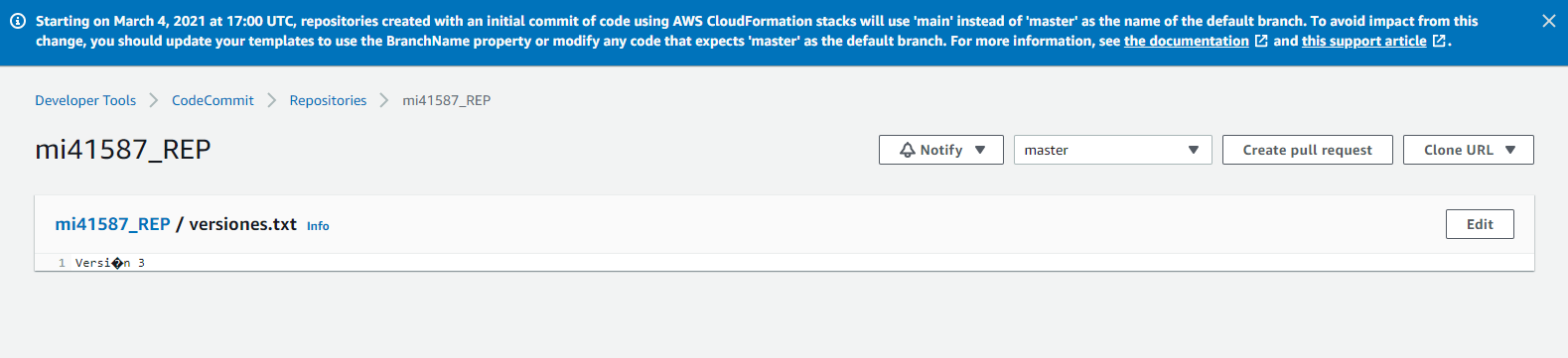
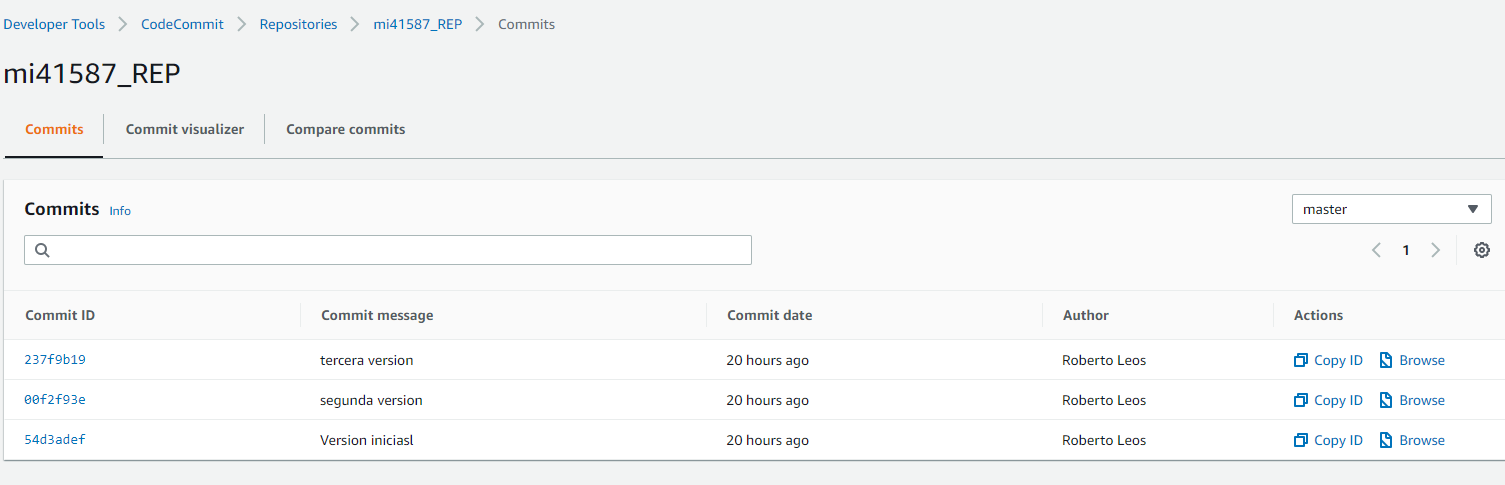


GIT HUB

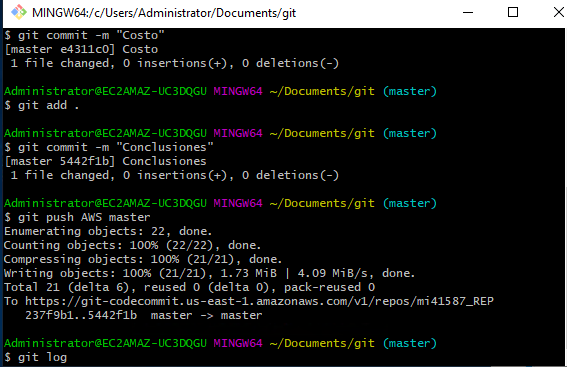
AWS CodeCommit

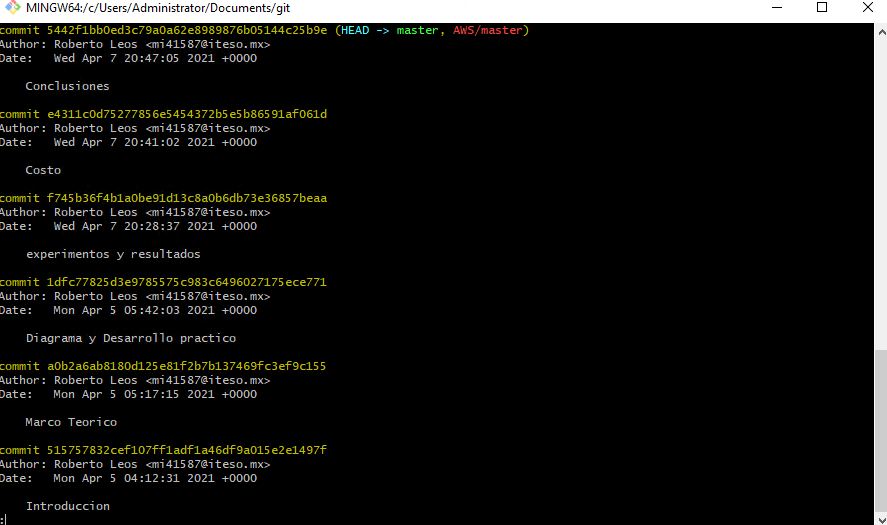
  


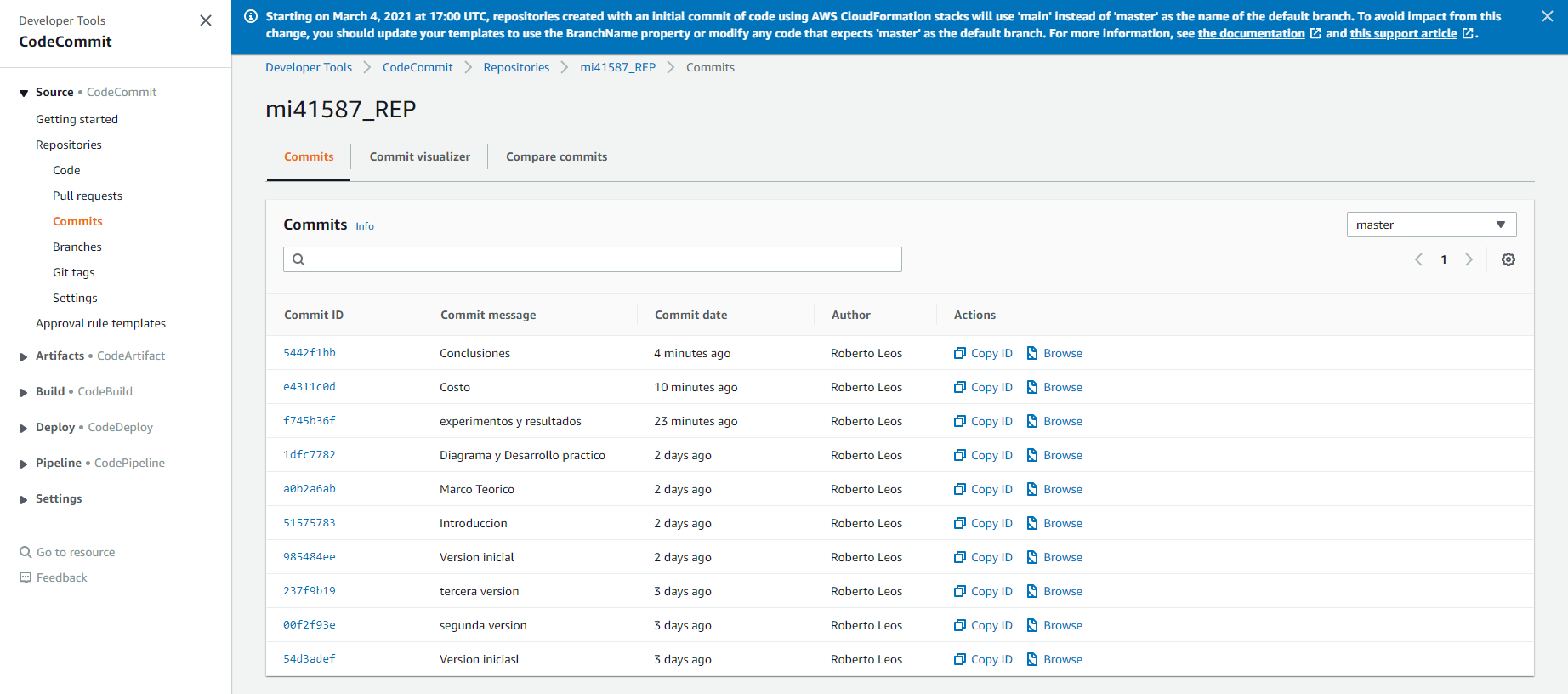
  
  


Git Ma

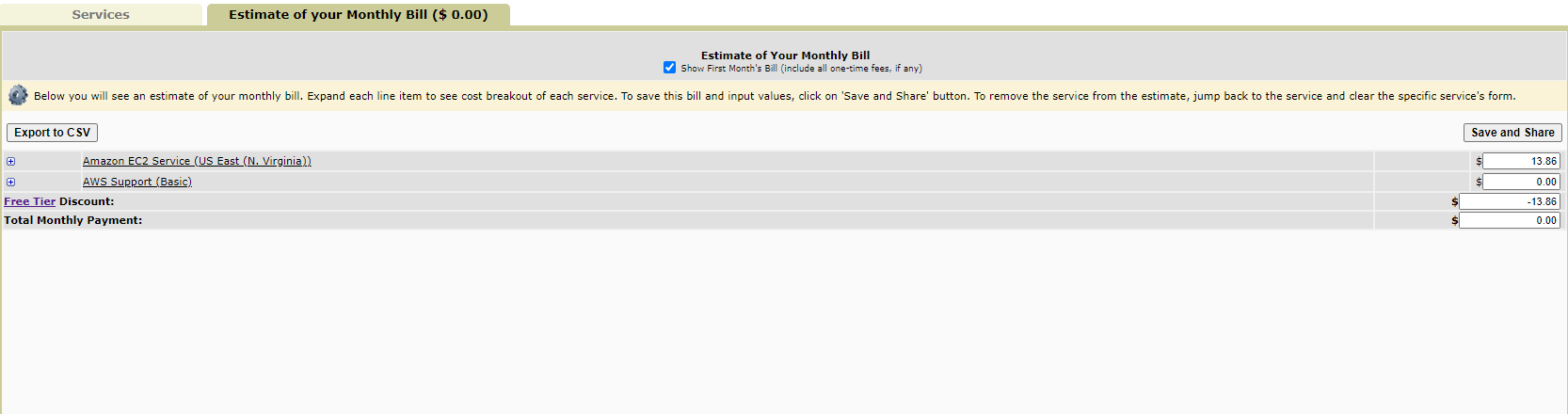
ster de la practica



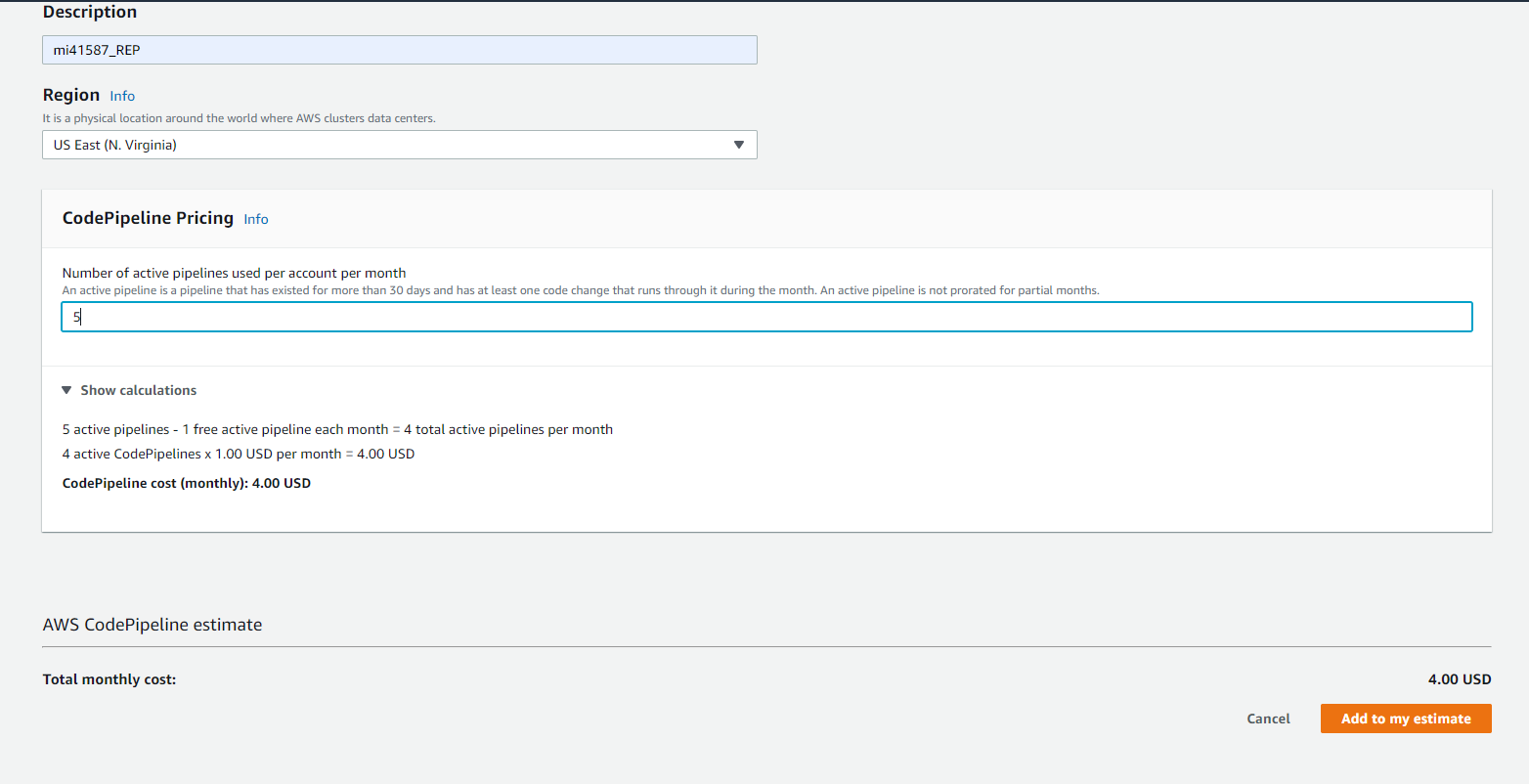




# Costo



encontré costo por el pipeline



# Conclusiones

El GIT es una opción muy sencilla de usar para el control de versiones no solo para códigos de programación si no también para todo tipo de documentos como procedimientos requerimientos y ver cuales fueron los cambios. Me agrado la opción hub pues puedes compartir los archivos fácilmente.

El codecommit está genial en AWS además que puedes manejar el ciclo de DevOps que es muy interesante el uso de eso usando el ciclo AWS CodeStar, CodePipeline, CodeCommit, CodeBuild, CodeDeploy.

# Bibliografía

|  |  |
| --- | --- |
| [1] | P. Agrawal1, «Devops, A New Approach To Cloud Development & Testing,» IEEE Xplore, KIET Group of Institutions, Ghaziabad, India, 2019. |
| [2] | AWS, «AWS,» [En línea]. Available: https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-welcome.html. |
| [3] | A. Rekognition, «AWS,» [En línea]. Available: https://docs.aws.amazon.com/rekognition/latest/dg/what-is.html. |
| [4] | B. Copeland, «artificial intelligence (AI),» Britannica ACADEMIC, [En línea]. Available: https://academic-eb-com.ezproxy.iteso.mx/levels/collegiate/article/artificial-intelligence/9711#article-contributors. |
| [5] | V. Srinivasan, Google Cloud Platform for Architects, packtpub, 2018. |