Escuela Colombiana de Ingeniería Julio Garavito

# Evaluation of a Blockchain-based solution for the security event management in IoT ecosystems

Trabajo Dirigido
Programa de Ingeniería de Sistemas

Director: Daniel Orlando Diaz López

Juan Camilo Velandia
Karen Paola Duran Vivas
2019-II

# Contenido

# Lista de Ilustraciones

# Lista de Tablas

# Resumen

Este proyecto es la continuación de un proyecto de grado “BSIEM-IoT: A blockchain-based and distributed SIEM for the Internet of Things”, su propósito es la realización de un prototipo mejorado de la solución BSIEM-IoT para lo cul se implementó una solución de seguridad que incluye una integración de eventos almacenados en la Blockchain y un SIEM desplegado en la nube que permite soportar escenarios de ciudades inteligentes. Para llevar a cabo este proyecto se implementaron algunas DAPP´s para lograr una mejor comprensión de su funcionamiento.

# Objetivo General

Evaluar una solución de gestión de eventos de seguridad que permita a un centinela recolectar y almacenar eventos en una red Blockchain para luego ser clasificados por sistemas SIEM conectados a los mineros dentro de la red.

## Objetivos Específicos

1. Entender las generalidades de la tecnología Blockchain identificando sus ventajas y desventajas, y sus aplicaciones.
2. Entender que es una DApp (Decentralized Application) y su diferencia con las aplicaciones convencionales además de aprender las bases del desarrollo en Blockchain.
3. Comprender la arquitectura de una solución de SIEM distribuido que utiliza centinelas IoT para recolección de eventos en ambientes IoT, específicamente la solución "BSIEM-IoT: A blockchain-based and distributed SIEM for the IoT" (Pardo, Ardila, & Díaz, 2019).
4. Proponer un conjunto de experimentos que incluyan diferentes casos de uso para la gestión de eventos de seguridad en entornos IoT distribuidos.

# Justificación

Blockchain es una tecnología que permite solucionar problemas actuales en diferentes escenarios de aplicación, por ejemplo, manufactura, transporte, gestión ambiental, entre muchos otros, gracias al modelo disruptivo que propone. De hecho, en el ámbito financiero Blockchain tiene gran cavidad a través de la criptomoneda Bitcoin, presentando un nuevo modelo del sistema financiero descentralizado, sin intermediarios y completamente transparente.

Las características de blockchain han permitido solucionar problemas de seguridad y confianza en la industria financiera y han permitido pensar en la aplicación que podría tener Blockchain en otros sectores como el de la ciberseguridad.

Se evidencia una oportunidad importante de explorar la aplicación de blockchain en aspectos de la ciberseguridad como la **gestión de eventos de seguridad en ambientes distribuidos**, ya que su inmutabilidad y consistencia hace imposible la modificación no autorizada y permite llevar una trazabilidad en escenarios donde participan múltiples actores como lo es un **entorno de IoT**.

# Productos

Para tener un mejor entendimiento sobre el funcionamiento de Blockchain, se decidió implementar varios prototipos de prueba antes de llegar al producto final. Los prototipos desarrollados fueron: La implementación de una DAPP de prueba de mensajería electrónica, la implementación de una DAPP de prueba de voto electrónico y la implementación de "Smart Contract-Based Access Control for the Internet of Things". Para mirar detalladamente el paso a paso a seguir para el desarrollo de cada uno de estos prototipos puede ingresar a la siguiente dirección URL: *https://github.com/jcamilovelandiab/BSIEMv2*, allí se encuentra toda la información correspondiente a este proyecto junto con la guía de implementación de los prototipos realizados.

A continuación, encontraremos una tabla de referencia, con la ubicación de cada uno de los productos realizados, tanto de los prototipos adyacentes a este proyecto como de la guía del producto final: "Instalación del nodo Ethereum en OSSIM".

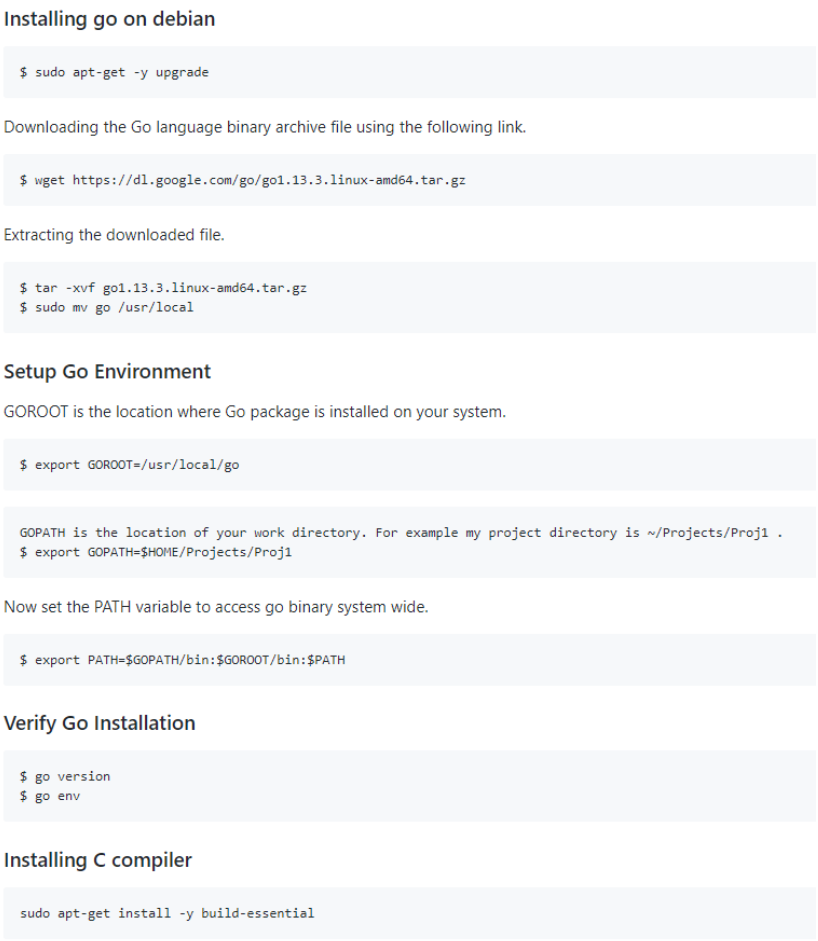
| | |
|---|---|
| Implementación de una DAPP de prueba de mensajería electrónica. | La información de este producto se encuentra en el siguiente link: *https://github.com/jcamilovelandiab/BSIEMv2/tree/master/ethereumdapp*.<br><br>Si desea ver la guía de implementación la puede encontrar en el siguiente enlace: *https://github.com/jcamilovelandiab/BSIEMv2/blob/master/ethereumdapp/Documents/Gu%C3%ADa%20implementaci%C3%B3n%20DAPP%20de%20mensajer%C3%ADa.pdf* **(Véase en el anexo 1. Guía de implementación DAPP de mensajería)**. |
| Implementación de una DAPP de prueba de voto electrónico. | La información correspondiente a este producto se encuentra en el siguiente link: *https://github.com/jcamilovelandiab/BSIEMv2/tree/master/election-dapp-master* **(Véase en el anexo 2. Guía de implementación DAPP de prueba de voto electrónico)**. |
| Implementación de "Smart Contract-Based Access Control for the Internet Things". | La información correspondiente a este producto se encuentra en el siguiente link: *https://github.com/jcamilovelandiab/BSIEMv2/tree/master/Smart%20Contract-Based%20Access%20Control%20for%20the%20Internet%20of%20things* |
| Instalación del nodo Ethereum en OSSIM y despliegue en la nube. | Cada uno de los pasos que se llevaron a cabo para la creación de este producto se describen en el siguiente documento. Si desea indagar un poco más puede consultar la guía de implementación de la instalación del nodo Ethereum que se encuentra en el siguiente enlace: *https://github.com/jcamilovelandiab/BSIEMv2/blob/master/SetupANode.md* |

*Tabla 1. Referencia de los productos desarrollados en el repositorio GitHub.*

Para empezar, se explicarán los pasos que se llevaron a cabo para la implementación de la infraestructura del BSIEM.

1. Lo primero que hay que hacer es instalar un nodo Ethereum en un servidor OSSIM 5.7.4, que en este caso es el sistema operativo que utilizaremos. Para esto, debemos asegurarnos de que todos los paquetes se encuentren actualizados, esto se hará mediante el siguiente comando: ***sudo apt-get update***.
   Vamos a realizar la instalación de geth con el fin de poder montar un nodo Ethereum en la máquina. Usamos geth porque vamos a trabajar con una blockchain de Ethereum. Los siguientes pasos son para instalar geth en nuestras máquinas. Sin embargo, hay un archivo standalone "./geth" en el siguiente repositorio ***https://github.com/jcamilovelandiab/BSIEMv2***. Eso significa que sólo necesitamos copiar el archivo geth a la carpeta de destino, y funcionará en Debian 8 x64bits. Si elegimos saltarnos los siguientes pasos referentes a la instalación de geth, simplemente clonamos el repositorio con el siguiente comando: "git clone https://github.com/jcamilo velandiab/BSIEMv2" y copiamos el archivo ".get".

   **Pasos para instalar geth:**
   Procedemos a clonar el repositorio de go-ethereum con el siguiente comando: ***git clone https://github.com/ethereum/go-ethereum.git***.
   Geth requiere la instalación de compiladores Go y C. Los pasos para instalarlo se encuentran especificados en la siguiente imagen:



**Installing go on debian**

```
$ sudo apt-get -y upgrade
```

Downloading the Go language binary archive file using the following link.

```
$ wget https://dl.google.com/go/go1.13.3.linux-amd64.tar.gz
```

Extracting the downloaded file.

```
$ tar -xvf go1.13.3.linux-amd64.tar.gz
$ sudo mv go /usr/local
```

**Setup Go Environment**

GOROOT is the location where Go package is installed on your system.

```
$ export GOROOT=/usr/local/go
```

```
GOPATH is the location of your work directory. For example my project directory is ~/Projects/Proj1 .
$ export GOPATH=$HOME/Projects/Proj1
```

Now set the PATH variable to access go binary system wide.

```
$ export PATH=$GOPATH/bin:$GOROOT/bin:$PATH
```

**Verify Go Installation**

```
$ go version
$ go env
```

**Installing C compiler**

```
sudo apt-get install -y build-essential
```

*Ilustración 1. Pasos para la instalación de Go y C*

Por último, procedemos a instalar geth:

***cd go-ethereum***

***make geth***

Para correr los nodos se puede hacer de la siguiente manera: build/bin/geth, claro está que el archivo ejecutable de geth puede moverse a cualquier carpeta del sistema y no se tendrán problemas.

2. El siguiente paso es instalar Nodejs.
   Necesitamos Nodejs para poder ejecutar los clientes de javascript. Usamos clientes de javascript para poder consumir la blockchain.

   Hay muchas maneras de instalar Node.js a través de apt. Una de ellas es usar una herramienta especialmente diseñada llamada nvm, que significa "Gestor de versiones de Node.js". Usando nvm, podemos instalar múltiples versiones autónomas de Node.js que nos permitirán controlar el entorno más fácilmente. Nos dará acceso a las versiones más recientes de Node.js bajo demanda, pero también nos permitirá dirigirnos a las versiones anteriores de las que nuestra aplicación pueda depender.

Al instalar Nodejs también NPM será instalado automáticamente. **NPM** significa "*Node Package Manager*" es el sistema de gestión de paquetes por defecto para Node.js, NPM se ejecuta desde la línea de comandos y maneja las dependencias para una aplicación, es decir, a través del cual se puede obtener cualquier librería escribiendo una línea de código, nos permite agregar dependencias de forma sencilla, distribuir paquetes y administrar el proyecto en general.

Los pasos de instalación de Nodejs los puede encontrar en el siguiente link ***https://www.digitalocean.com/community/tutorials/how-to-install-node-js-on-debian-8***, o puede ingresar directamente a la URL del repositorio del proyecto: ***https://github.com/jcamilovelandiab/BSIEMv2/blob/master/SetupANode.md*.**

3. Por último, se explicará cómo se instalan y configuran los nodos de la blockchain. Primero se creará un nuevo folder, en este caso lo llamamos myDataDir, y dentro de esta carpeta de crea un archivo JSON llamado génesis con los valores: chainId, difficulty, gasLimit y alloc. Este archivo debe contener la siguiente información:

```
{
  "config": {
     "chainId": 1994,
     "homesteadBlock": 0,
     "eip150Block": 0,
     "eip155Block": 0,
     "eip158Block": 0,
     "byzantiumBlock": 0
  },
  "difficulty": "400",
  "gasLimit": "2000000",
  "alloc": {
     "put_the_address_of_the_wallet" : {
         "balance": "1000000000000000000000000"
     }
  }
}
```

*Ilustración 2. Contenido del archivo Genesis*

El archivo "./genesis" lo podemos encontrar en la raíz del repositorio ***https://github.com/jcamilovelandiab/BSIEMv2***.
Ahora se inicializa la blockchain con el siguiente comando:
***./geth --datadir ./myDataDir init ./genesis.json***
Con la ejecución del comando anterior se creará una carpeta vacía. Ahí es donde se va a guardar la información de la blockchain.

4. Para el siguiente paso se necesitan crear wallets o billeteras electrónicas, esto se puede hacer a través de diferentes herramientas como lo son: MyEthereumWallet que es una billetera popular Ethereum del lado del cliente en línea, otra opción es MetaMask que es otra billetera Ethereum ampliamente utilizada en el navegador, y existen muchas otras opciones como: Nethereum. En Ethereum blockchain las wallets se almacenan en un formato especial de cifrado conocido como "UTC / JSON Wallet (archivo de almacenamiento

de claves)", este es el formato de archivo de la wallet utilizado en geth y Parity (Estos dos son las principales implementaciones de protocolo para Ethereum).

Pasos para crear una billetera en Ethereum con Myetherwallet.

- Visitamos el sitio https://www.myetherwallet.com/.
- Le damos click sobre crear una nueva billetera. (Create a new wallet), después myetherwallet nos dará una breve introducción a lo que es una billetera blockchain, es importante que leamos esta introducción. Simplemente le damos continuar a todo.
- Ya después de pasar de la introducción, vamos a crear una nueva billetera por medio del archivo keystore (by keystore file), es importante seleccionar esta opción porque vamos a utilizar esos archivos para copiarlos al directorio de la blockchain "myDataDir".
- Le asignamos una contraseña, le das en continuar y por último descargas el archivo.
- Una vez descargado el archivo vamos a acceder a la billetera usando el archivo keystore que acabamos de descargar.
- Accedemos por medio del archivo, e ingresamos la contraseña de la billetera que le asignamos anteriormente.
- En el panel de myetherwallet podemos ver el balance que tiene de la billetera, incialmente tenemos 0 eth, y también vemos la dirección de la billetera que es un número hexadecimal.

5. Ahora vamos a replicar el proceso de la creación de la billetera virtual. Creamos n billeteras virtuales para n mineros, esto con el fin de que cada minero tenga su propia billetera virtual para poder minar, pues al momento de minar bloques se irán descontando ethers a la billetera asociada al minero. Después, copiamos todos los archivos UTC creados a una carpeta llamada /keystore en "./myDataDir".

6. Antes de ejecutar el nodo tenemos que deshabilitar el firewall, ya que el servidor OSSIM no tiene habilitados los puertos por seguridad, y necesitamos habilitar esos puertos o deshabilitar el firewall para que en un futuro podamos leer la blockchain e interactuar con ella. Vamos a comunicarnos con la blockchain a través de remix.

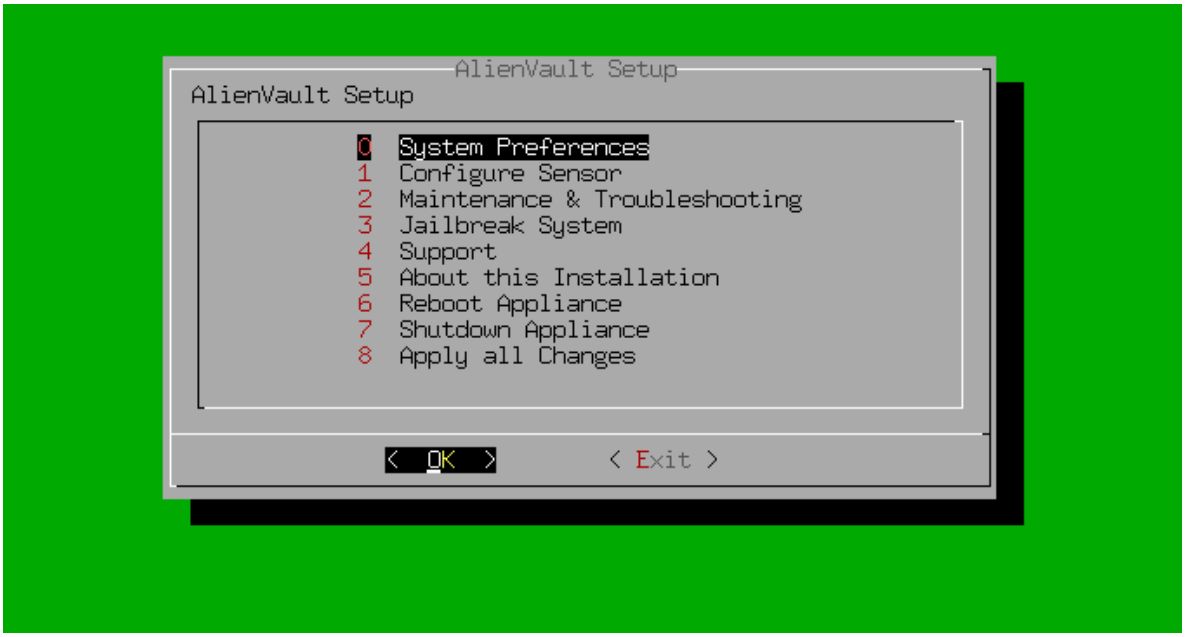



*Ilustración 3. Menú principal de la máquina virtual OSSIM.*

Para deshabilitar el firewall seguimos lo siguientes pasos: Vamos al menú principal de la máquina, hacemos clic en Preferencias del sistema, luego en Configurar red, luego en AlientVault Firewall, y finalmente seleccionamos la opción "no" para deshabilitar el firewall y hacemos clic en ok. Volvemos al menú principal y aplicamos todos los cambios. Volvemos al menú principal y aplicamos todos los cambios.

7. Ahora procedemos a ejecutar el nodo, para esto se debe habilitar la conexión RCP y la conexión WebSocket. El siguiente comando se usa solamente para los mineros (no para los centinelas), como se puede observar hay un –port, --rpcport y –wsport, para este proyecto se utilizaron los siguientes puertos respectivamente: 3000, 3001 y 8545. Si desea, puede utilizar otros puertos diferentes.

```
$ ./geth --datadir [path to data directory] --syncmode 'full' --port [entry port]
--rpc --rpcaddr [ip address of node] --rpcport [rcp port] --rpccorsdomain "*"
--ws --wsaddr [ip address of node] --wsport [web socket port] --wsorigins "*"
--networkid [network id] console --unlock [address of wallet] --allow-insecure-unlock
```

*Ilustración 4. Comando para mineros*

8. Para desplegar los Smart Contracts, se utilizará Remix como herramienta. Remix es una poderosa herramienta de código abierto que nos ayuda a escribir contratos de Solidity (Lenguaje de programación) directamente en el navegador, Remix también admite pruebas, depuración e implementación de contratos inteligentes. Para acceder a Remix, se puede hacer mediante el siguiente enlace [http://remix.ethereum.org/](http://remix.ethereum.org/).

Algo importante a tener en cuenta es que se necesita que los mineros se estén ejecutando cuando se realice el despliegue del Smart Contract, para inicializar un minero se utiliza el siguiente comando: ***miner.start()***. El minero puede detenerse después de que se haya generado el primer bloque, de lo contrario se continuarán extrayendo otros bloques para mantener la cadena de bloques.

9. Con el fin de tener una solución de seguridad para IoT integrada en la nube que pueda asegurar diferentes tipos de servicios IoT, procedemos a desplegar el servidor OSSIM en la nube, para esto debemos ir a la opción *"Explore online Demo"* del siguiente link: https://cybersecurity.att.com/products/usm-anywhere y diligenciar todos los datos para solicitar el demo de USM, que corresponde a la versión comercial de OSSIM. Posterior a esto, llegará un email al correo que se haya diligenciado anteriormente, el cual contiene un "*trial authentication code*", similar a la siguiente imagen.

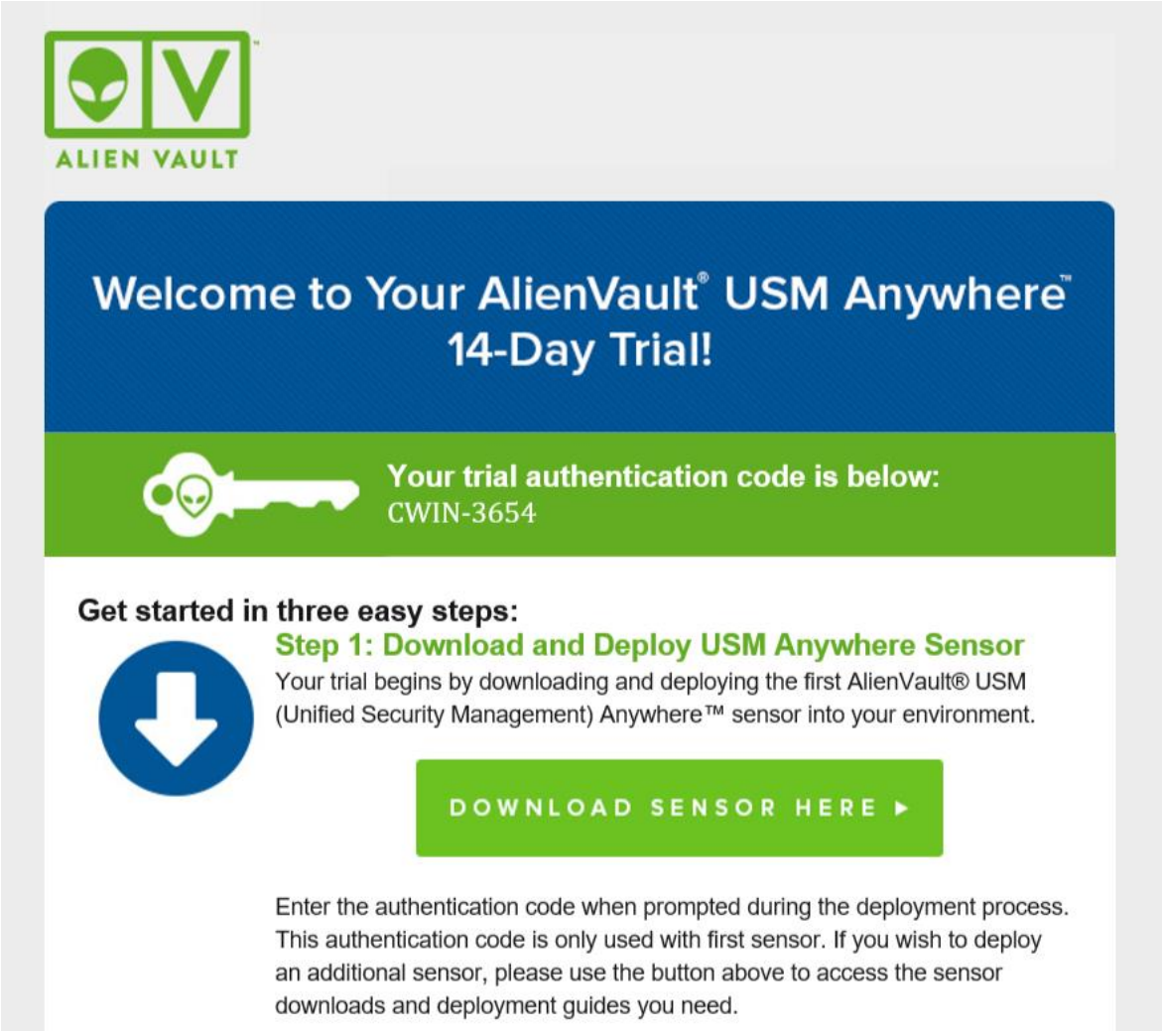



*Ilustración 5. Ejemplo del mensaje de autenticación*

10. Procedemos a dar click en el botón de "DOWNLOAD SENSOR HERE", nos va a aparecer una página similar a esta.

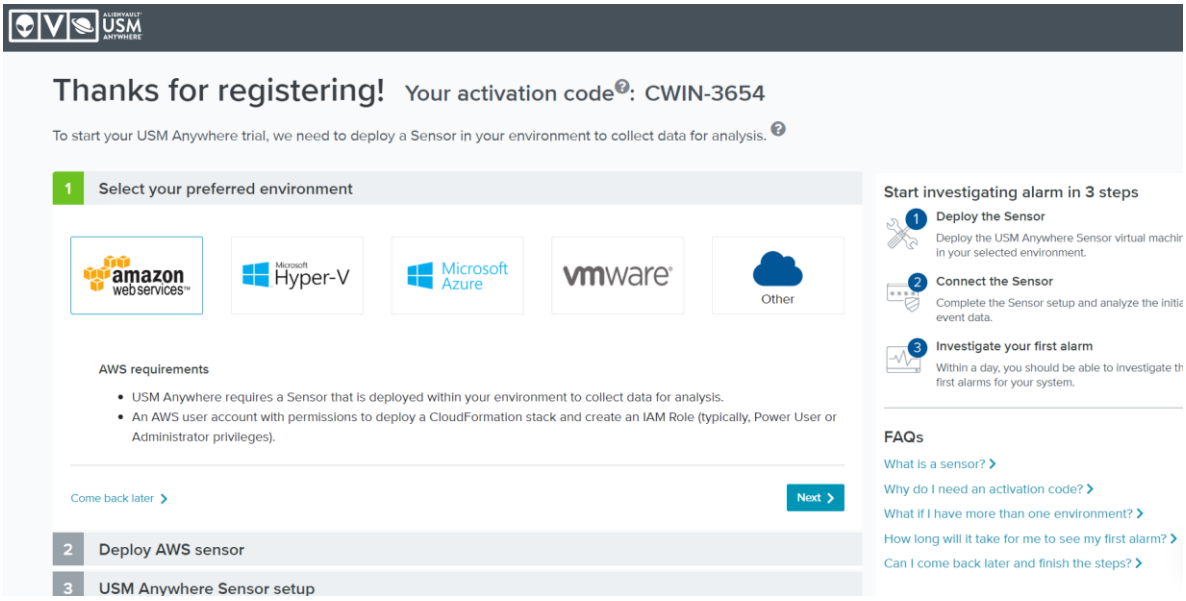



*Ilustración 6. Inicio de la página de descarga*

11. Podemos elegir cualquiera de las opciones, pero en este caso nosotros decidimos irnos por Amazon Web Services, porque ya tenemos una cuenta de AWS student que da Amazon para estudiantes de manera gratuita. Luego, damos click en siguiente y procedemos a seleccionar la región para desplegar el sensor.

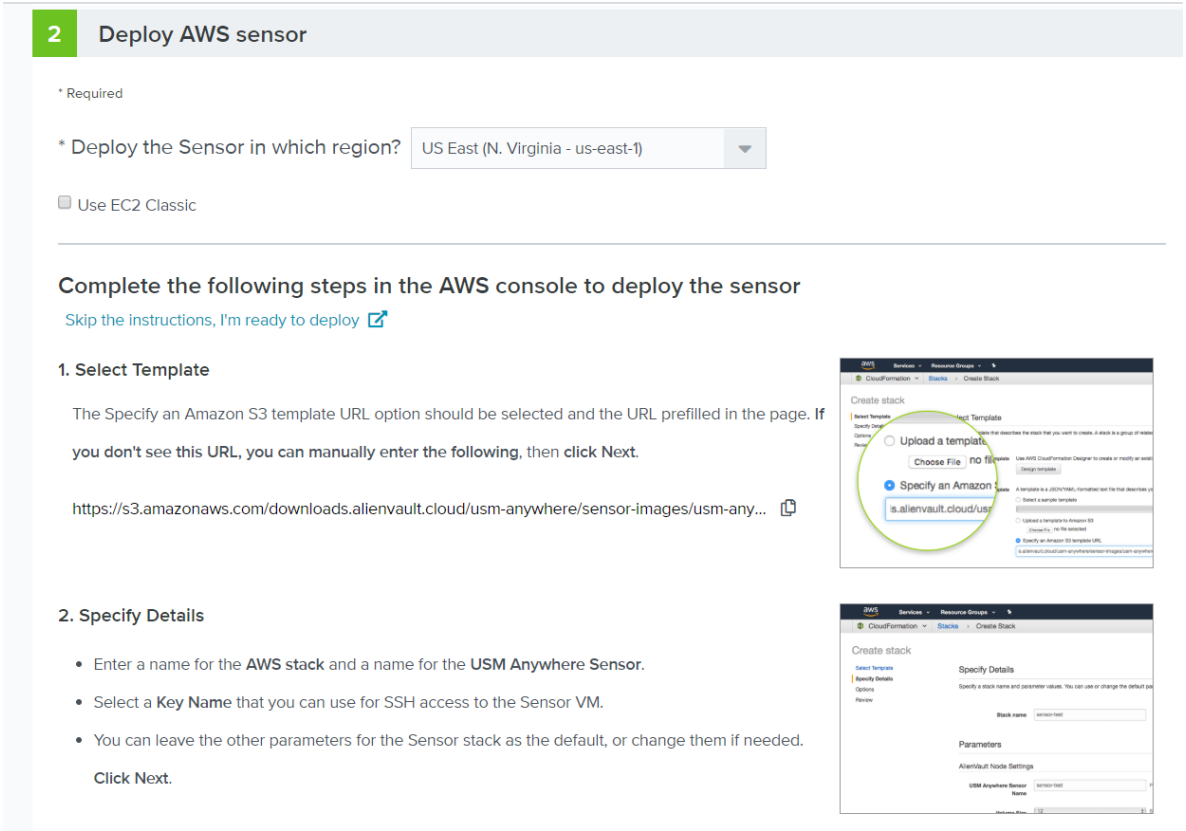



*Ilustración 7. Sección para elegir la región en donde se desplegará el sensor*

12. Vamos a la consola de AWS, y en la parte de buscar servicios ingresamos "*CloudFormation*", al ir al servicio nos aparecerá algo similar a esto.

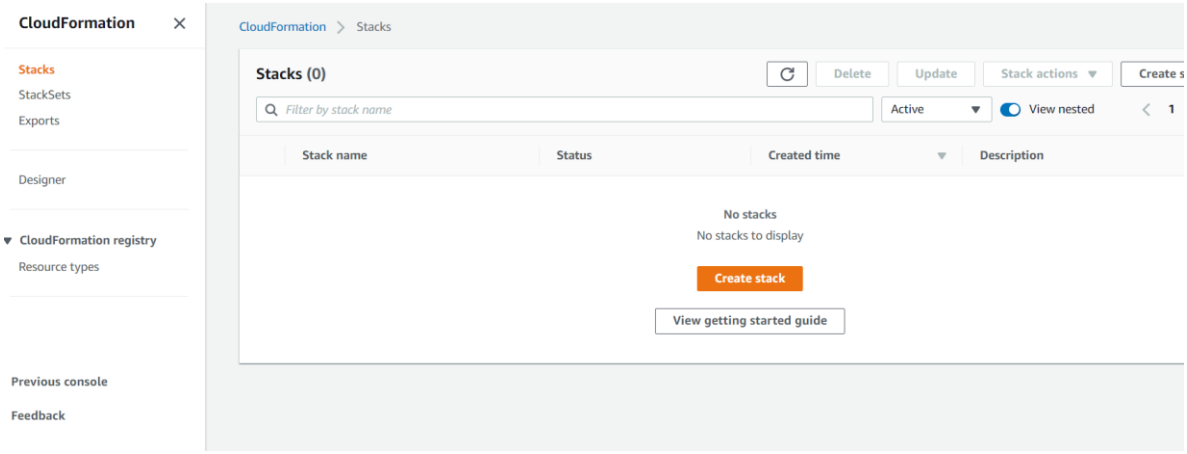



*Ilustración 8. Imagen de guía para CloudFormation*

13. Hacer click sobre "*Create Stack*".
14. Aparecerá un formulario para crear una stack, AWS nos pedirá llenar el campo Amazon S3 URL. Copiamos y pegamos la URL que nos aparecía en la página de AlientVault.

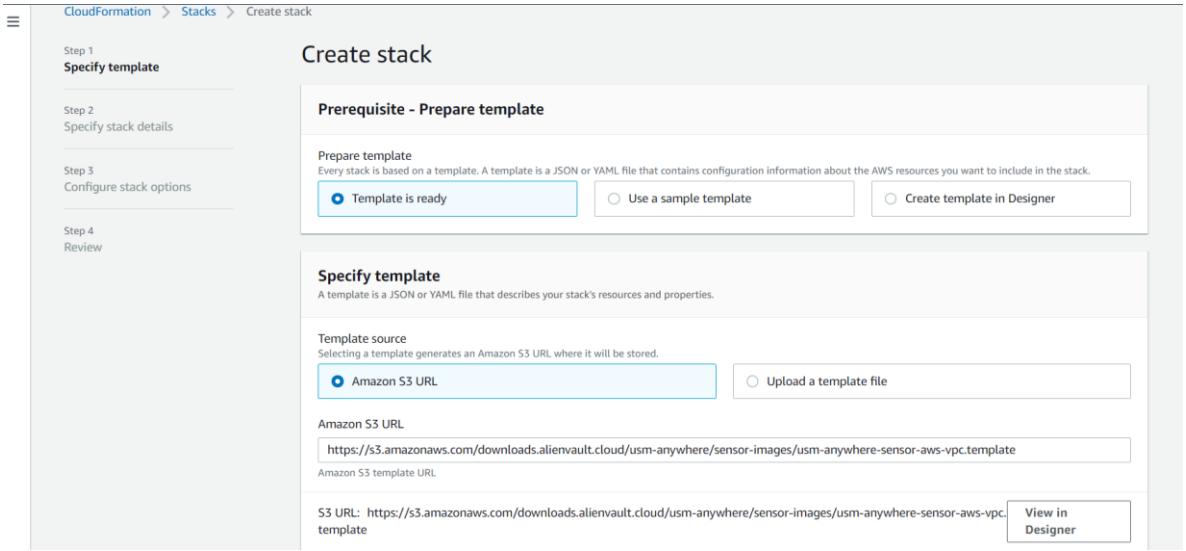



*Ilustración 9. Formulario para crear una stack*

15. Le pondremos un nombre al Stack, y los completamos los parámetros. En este caso, se crearon un par de llaves en AWS para el sensor, sin embargo, se puede utilizar un par de llaves ya existentes. EL par de llaves nos permitirá el acceso a nuestro USM Sensor a través de SSH.

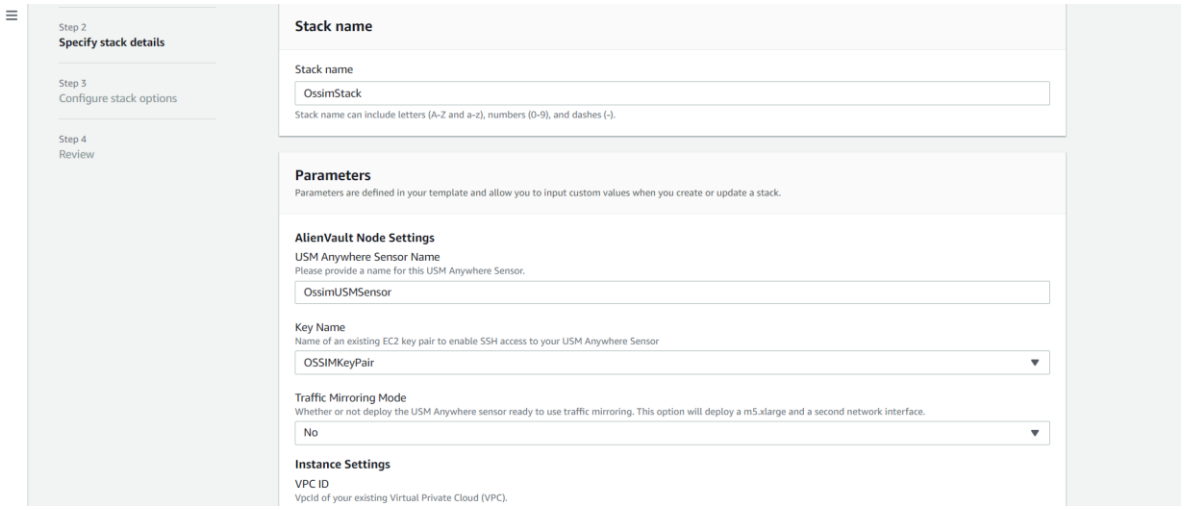



*Ilustración 10. Llenando los parámetros del formulario*

16. Luego seleccionamos la VPC por defecto, y elegimos una subred. También habilitaremos una IP pública para acceder a la página a través de una IP. Damos click en siguiente.

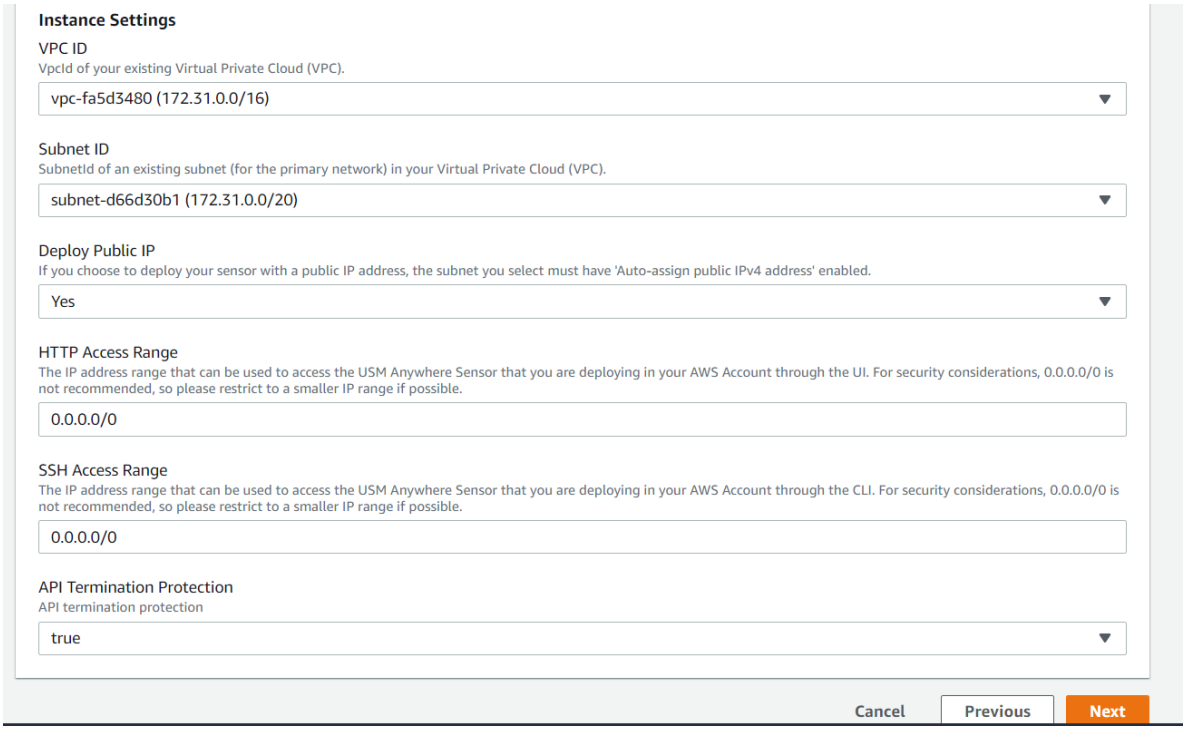



*Ilustración 11. Configuraciones de VPC, subred, IP*

17. En el paso de configurar las opciones de la Stack (Configure stack options), damos click en siguiente.
18. Revisaremos que los campos de la pila que vamos a crear estén correctos. Y creamos finalmente la pila.

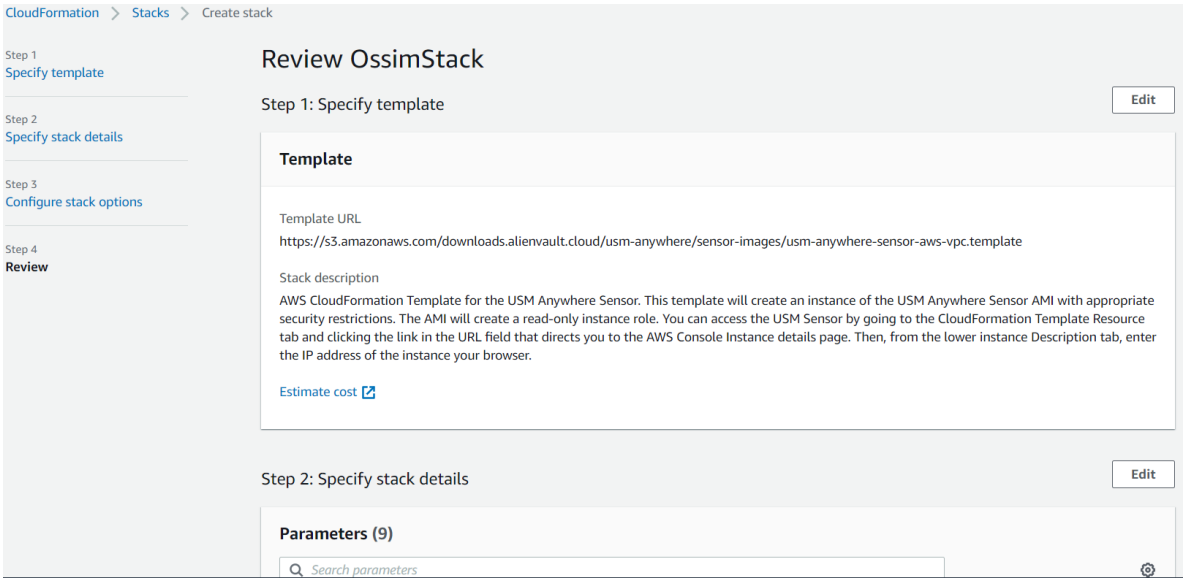



*Ilustración 12. Revisión de campos de pila I*

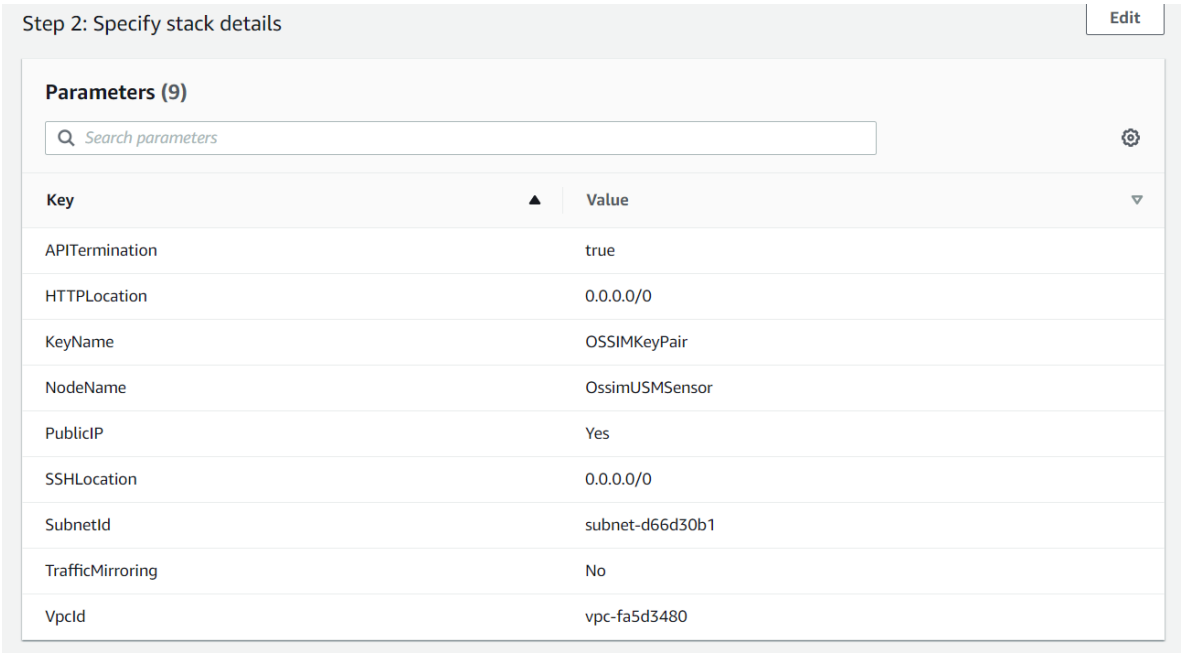



*Ilustración 13. Revisión de campos de pila II*

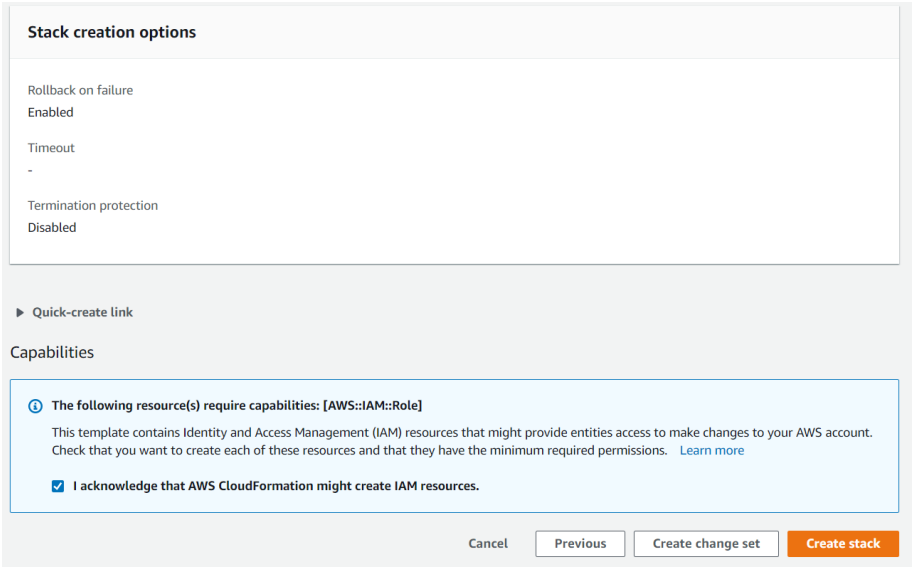



*Ilustración 14. Revisión campos de pila III*

19. Aparecerá como estado que se está creando el stack, tomará un par de minutos la creación de esta.

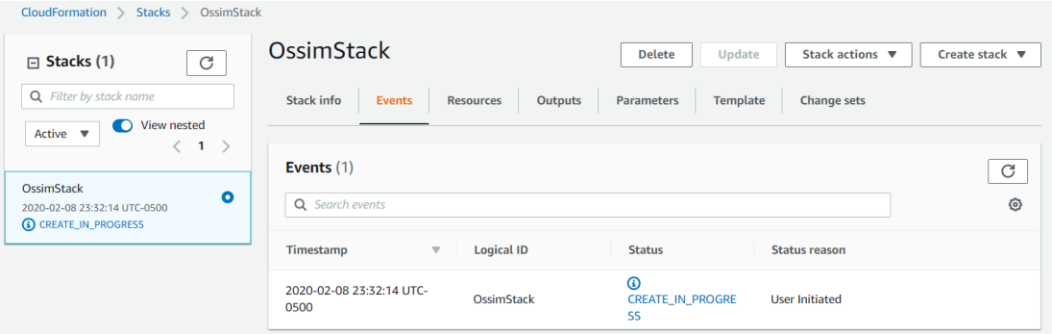



*Ilustración 15. Estado de creación de la pila*

20. Una vez terminada la creación del stack, aparecerán los siguientes eventos en el tablero de eventos.

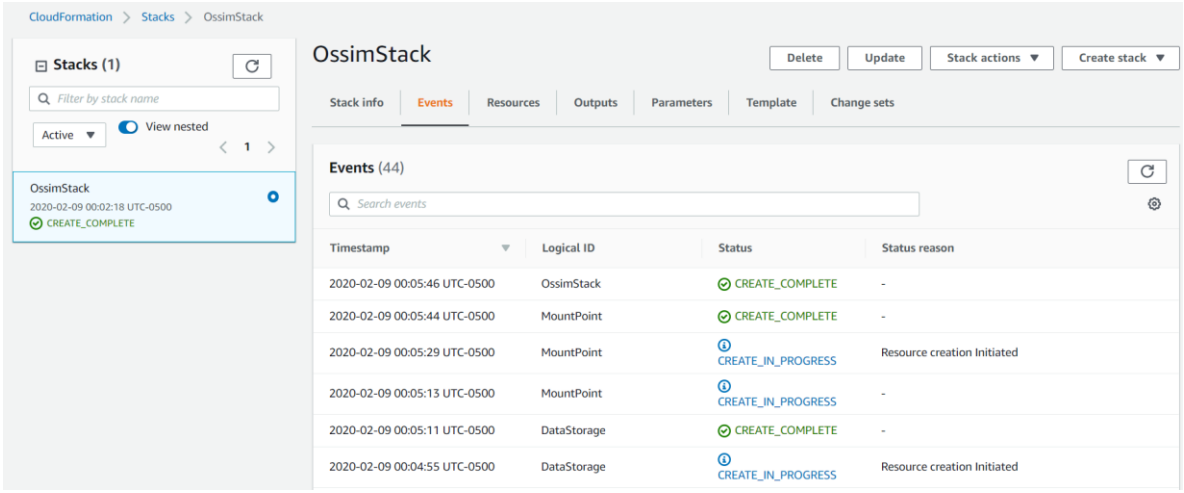



*Ilustración 16. Eventos tras la creación de la pila*

21. Nos dirigimos entonces, a la ventana de Outputs y entramos a la IP pública.

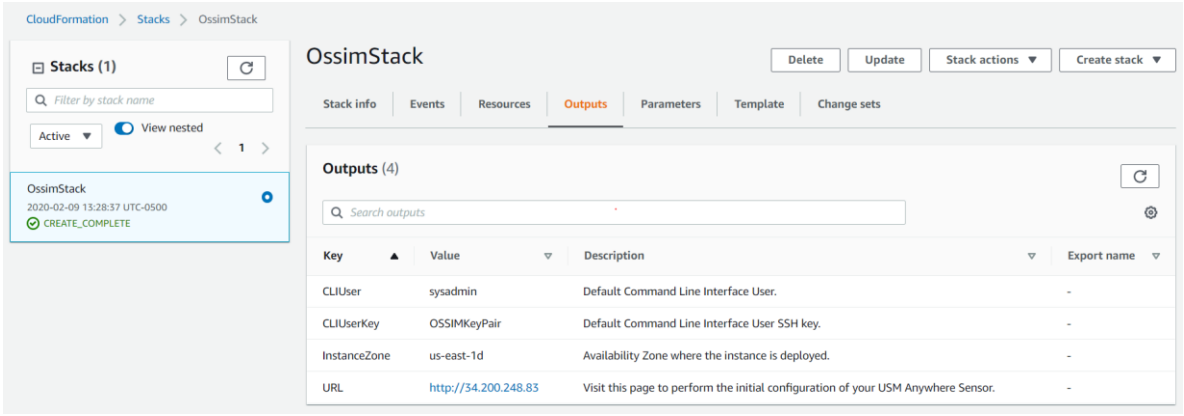



*Ilustración 17. Sección de Outputs*

22. Una vez en la página web, le colocamos un nombre al sensor e ingresamos una descripción de éste. También nos pedirá la llave que AlientVault nos envió al correo desde un inicio. Le damos click en “Start Setup”.

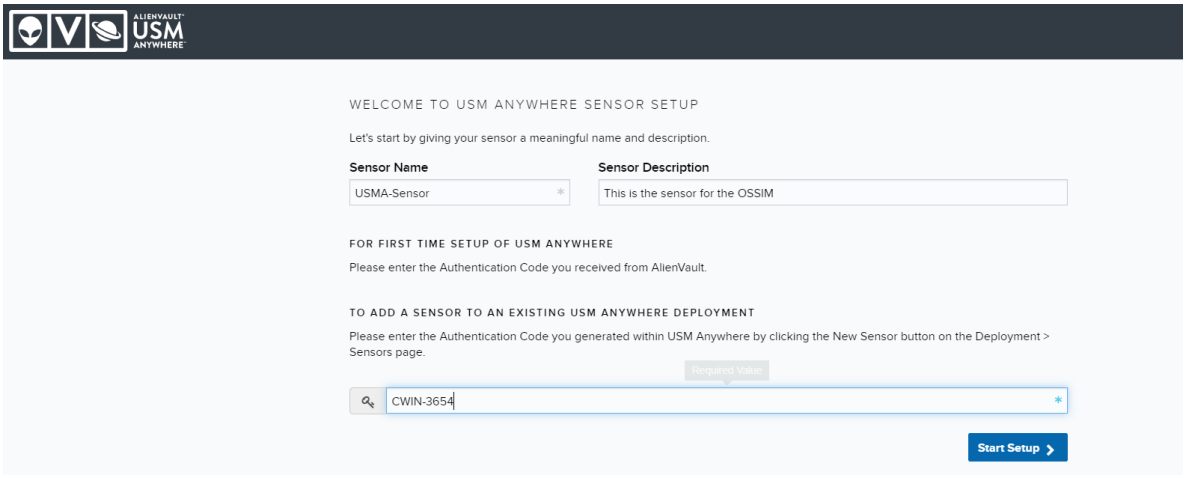



*Ilustración 18. Página de inicio USM*

23. Ahora, esperaremos a que se configure el sensor.

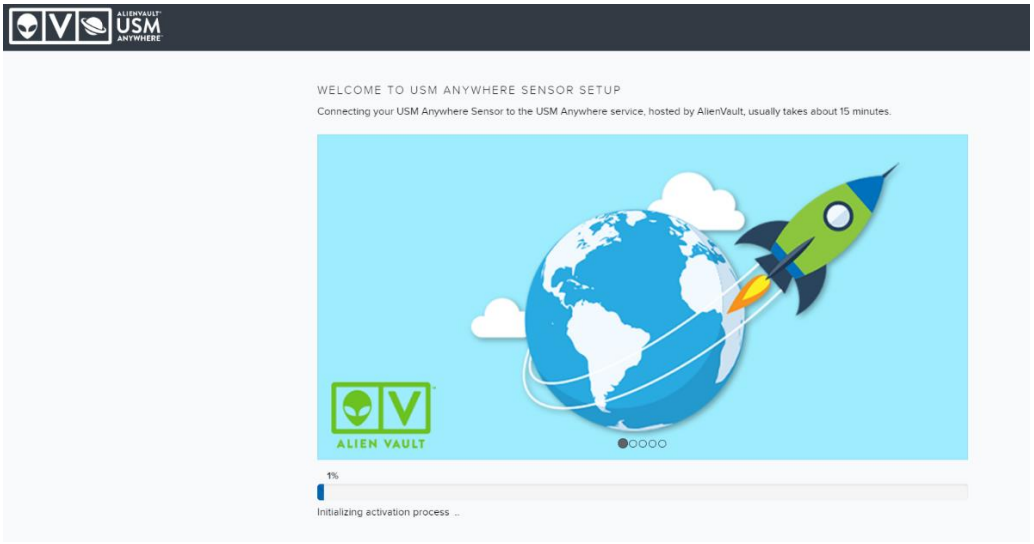



*Ilustración 19. Configuración del sensor*

24. Después de que se haya inicializado, damos click en "*Click Here*" para continuar con el proceso.

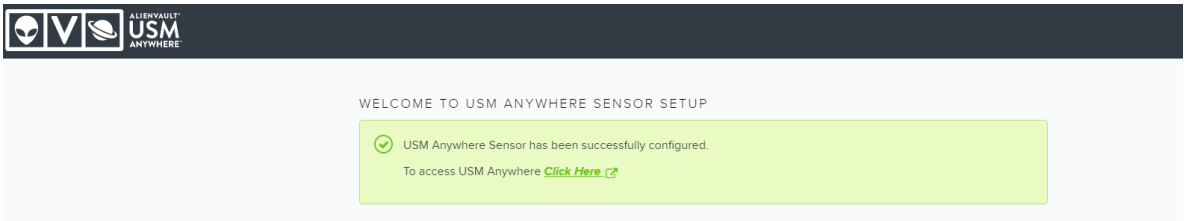



*Ilustración 20. Inicialización de USM*

25. Nos va a redirigir a una página web en el que nos va a pedir las credenciales. Ingresamos correo y contraseña.

26. Y ahora podremos ver el panel del sensor.

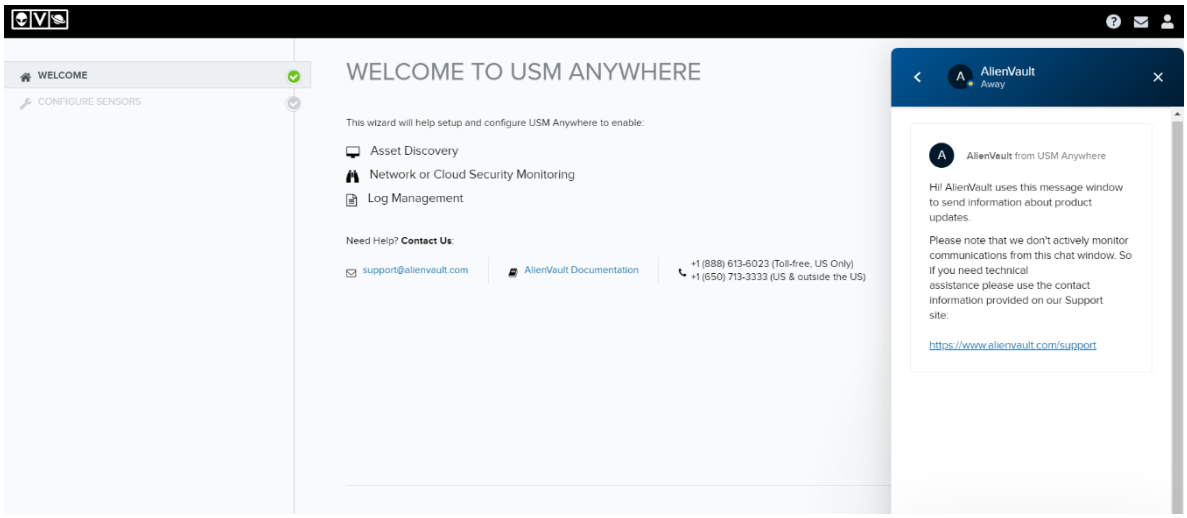



*Ilustración 21. Visualización del página inicial*

27. En la esquina inferior derecha hay un botón de "*Getting started*", se deberán seguir los pasos mostrados en esta página.
28. Finalmente, después de haber completado la configuración del sensor, nos aparecerá un panel de control similar al siguiente.

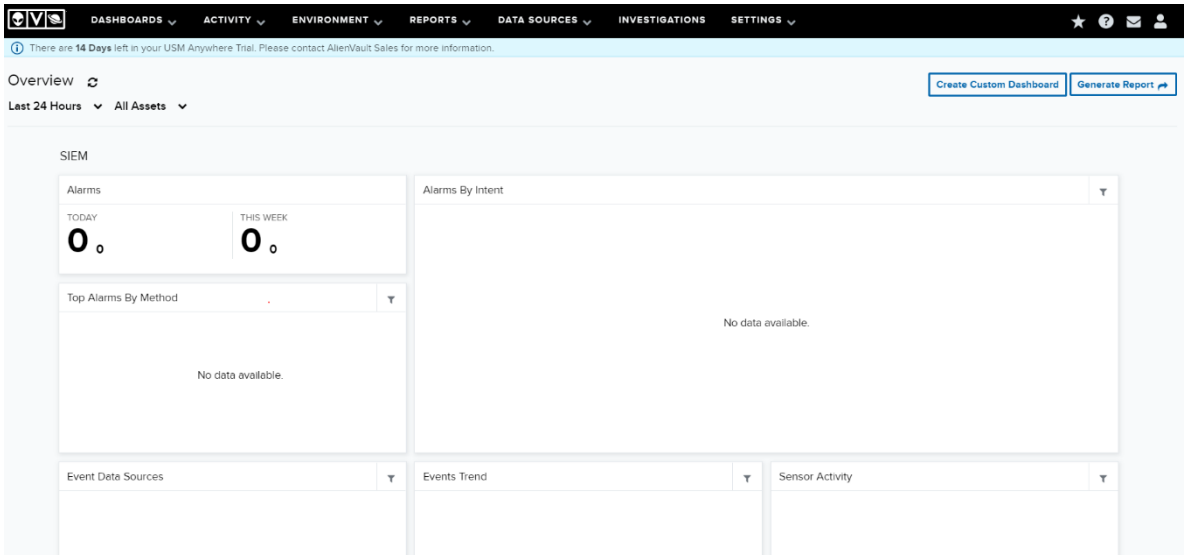



*Ilustración 22. Panel de control*

En este punto ya se tiene desplegado el servidor USM en la nube, el cual nos permitirá gestionar eventos de seguridad. Sobre este servidor se pueden realizar los pasos mencionados en la sección "Instalación del nodo Ethereum en OSSIM", y de esta forma poder lograr que el servidor USM incorpore las funciones de un minero Ethereum.

# Problemas y Soluciones

1. **Problema:** Familiarización con los conceptos de blockchain, DAPP's, contratos inteligentes. **Solución:** Indagaciones sobre el tema, pruebas de implementación en estos temas para poder tener una mejor comprensión de su funcionamiento.

2. **Problema:** Réplica del ambiente de pruebas del BSIEM-IoT.

   Hubo problemas con las dependencias que necesitábamos para el proyecto, también con el sistema operativo ya que el OSSIM que proporciona AlienVault tiene muchas restricciones en la instalación de dependencias requeridas por Geth. Además, montar un minero estable en una Raspberry PI3 con un OS Ubuntu Mate era demasiado complejo instalar todas las dependencias que necesitábamos, la mayoría de los pasos que encontramos en internet no sirven.

   **Solución**: Se construye la documentación respectiva de instalación Geth en un sistema operativo Debian igual al utilizado por OSSIM 5.7.4. Estos pasos quedaron documentados en la sección "Instalación del nodo Ethereum en OSSIM" del presente documento.

3. **Problema:** Extensión del BSIEM de un ámbito de smarthome a uno de Ciudades Inteligentes.

   **Solución:** Migración a la nube, debido a inestabilidades en la máquina virtual en donde se tenía el BSIEM la solución encontrada fue migrar el BSIEM a una instancia en la nube teniendo en cuenta también que en este escenario se pueden manejar modelos más robustos ya que nos proporcionan escalabilidad, ideal para escenarios de ciudades inteligentes.

# Conclusiones

I. La solución implementada posee una característica de resiliencia ante ataques distribuidos evitando que haya un único punto de ataque y que la infraestructura crítica a salvaguardar quede desprotegida.

II. La escalabilidad y auditoría demostradas en la solución implementada permiten determinar que esta puede ser aplicable a escenarios de ciudades inteligentes compuestos por múltiples subsistemas (transporte, ambiente, seguridad, gobierno, etc) con requerimientos de seguridad abordables por parte de una blockchain.

# Bibliografía

https://www.alienvault.io/product/onboarding?activationKey=CBGH-8302&utm_medium=Email&utm_source=Transactional&utm_content=ProductTrial&utm_campaign=USMA_Activation

https://github.com/andresvega82/SIEM-IoT/tree/master/Software

https://cybersecurity.att.com/documentation/usm-appliance/plugin-management/plugin-fundamentals.htm

https://cybersecurity.att.com/documentation/usm-appliance/events/event-details-fields.htm

http://mdlval.blogspot.jp/

https://github.com/anfepar/BSIEM

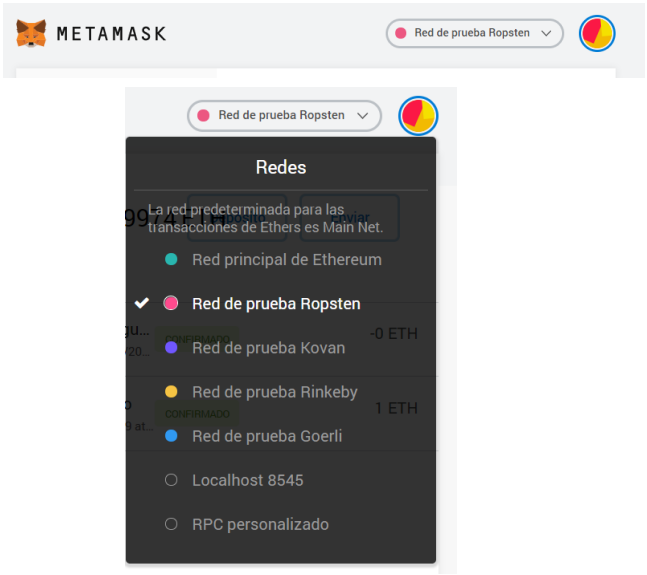
# Anexos

## Anexo 1

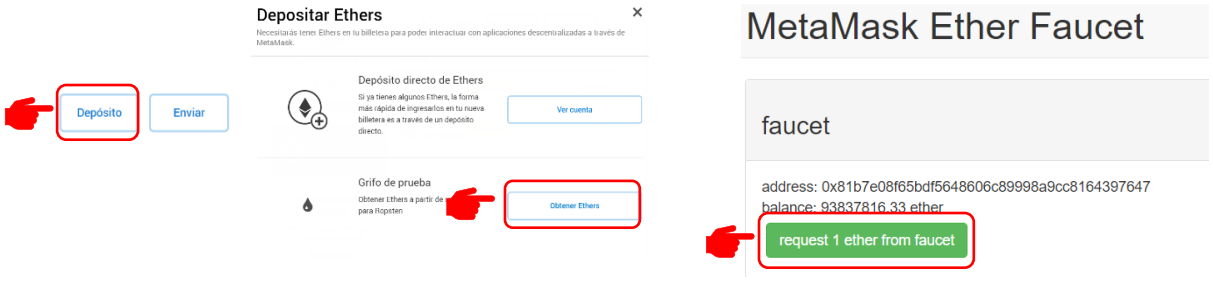
## Guía implementación DAPP de mensajería

### Instalación METAMASK

Para poder comenzar con la implementación necesitaremos descargar en nuestro navegador la extensión de Metamask que será nuestra wallet, lo puede hacer a través del siguiente link https://metamask.io/. Después de esto, necesitaremos cambiar la opción de red que se encuentra en la parte superior, por la red de prueba Ropsten.



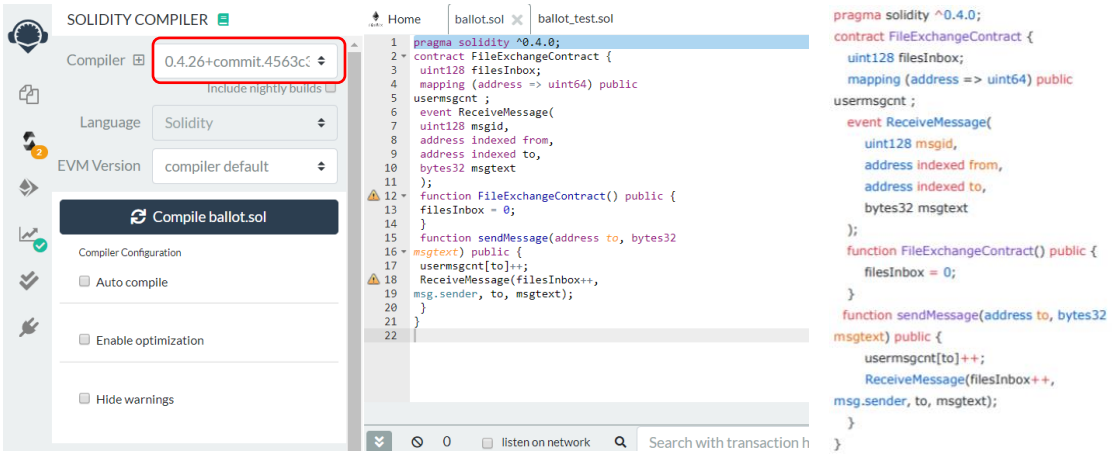

Ahora es necesario solicitar un Ether para poder realizar las transacciones. Para esto damos click en el botón de *"Depósito"*, y luego click en *"Obtener Ethers"*, por último, damos click en *"request 1 ether form faucet"* y ya tendremos nuestro primer en Ether en la billetera virtual.



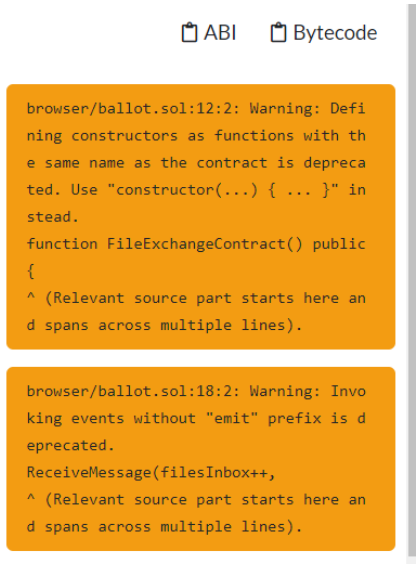
## Compilar el código en REMIX

El siguiente paso es abrir el IDE de Solidity online, llamado REMIX. https://remix.ethereum.org/. En el IDE se copiará el código de la imagen.

Algo importante a tener en cuenta es la versión del compilador, para este caso la versión que asumimos que es la correcta es la 0.4.26+commit.

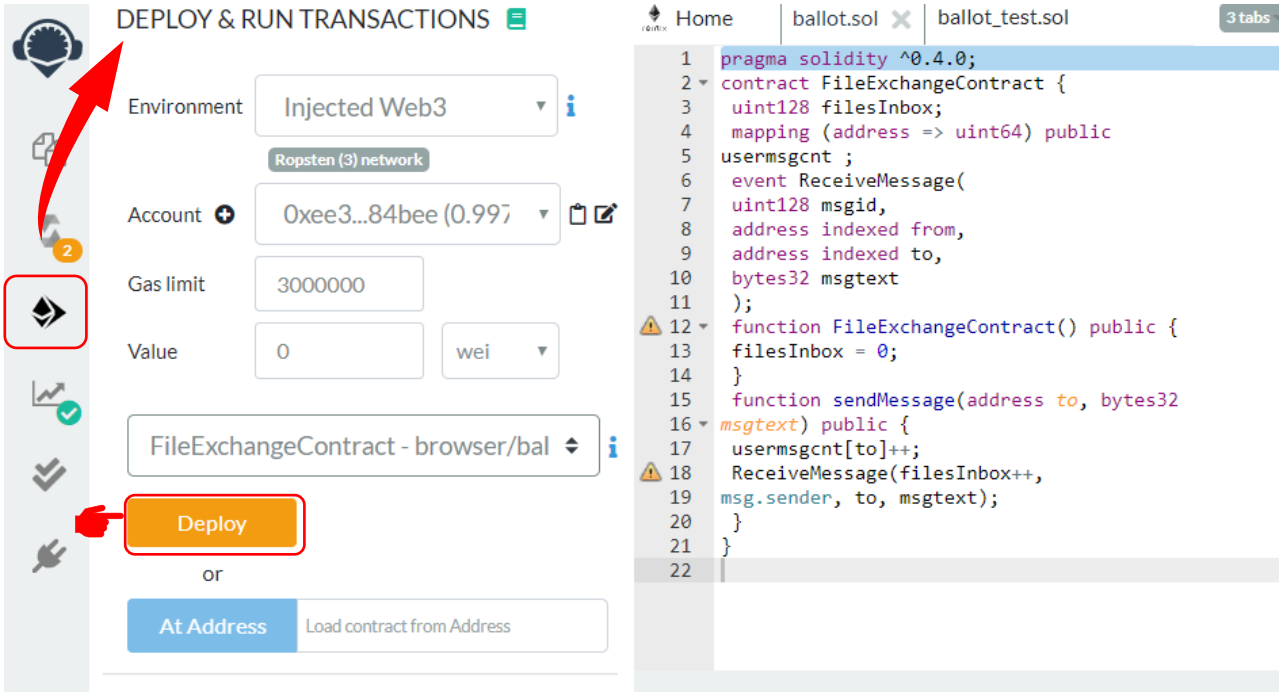



Y se compila el código. Aparecerán algunas alertas en color amarillo, pero no es razón para preocuparse.

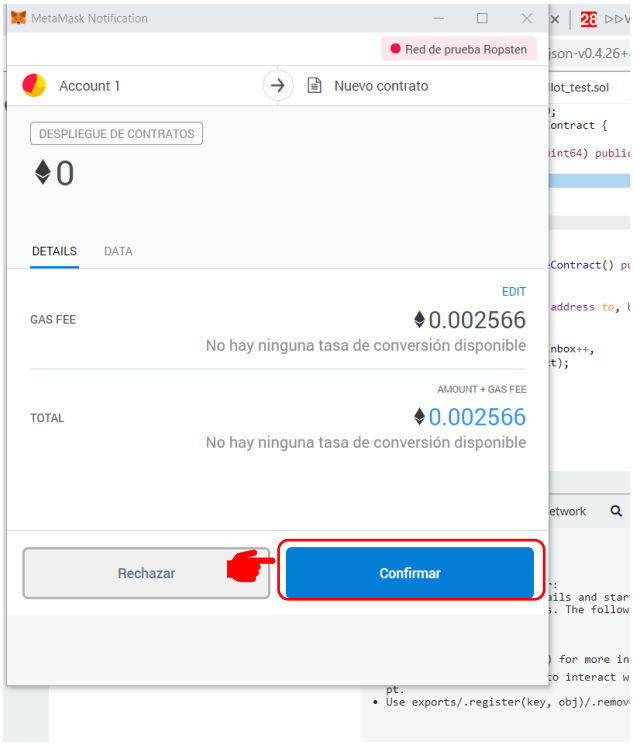



## Desplegar el contrato

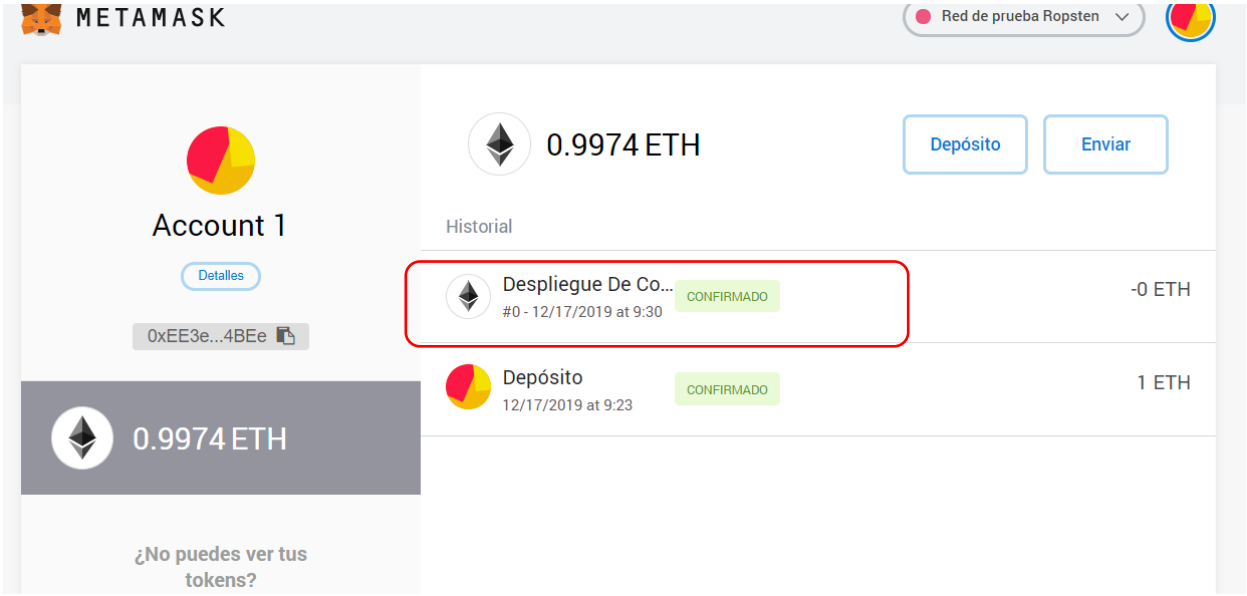
El siguiente paso será el despliegue del contrato, para esto nos vamos a la sección de *"Deploy & run transaction"* ubicada en el menú de la parte lateral izquierda. En *"Environment"* escogemos la opción de **Injected Web3** y automáticamente nos debería reconocer la cuenta de Metamask en la opción de *"Account"*. Ahora seleccionamos el botón de *"Deploy"*.

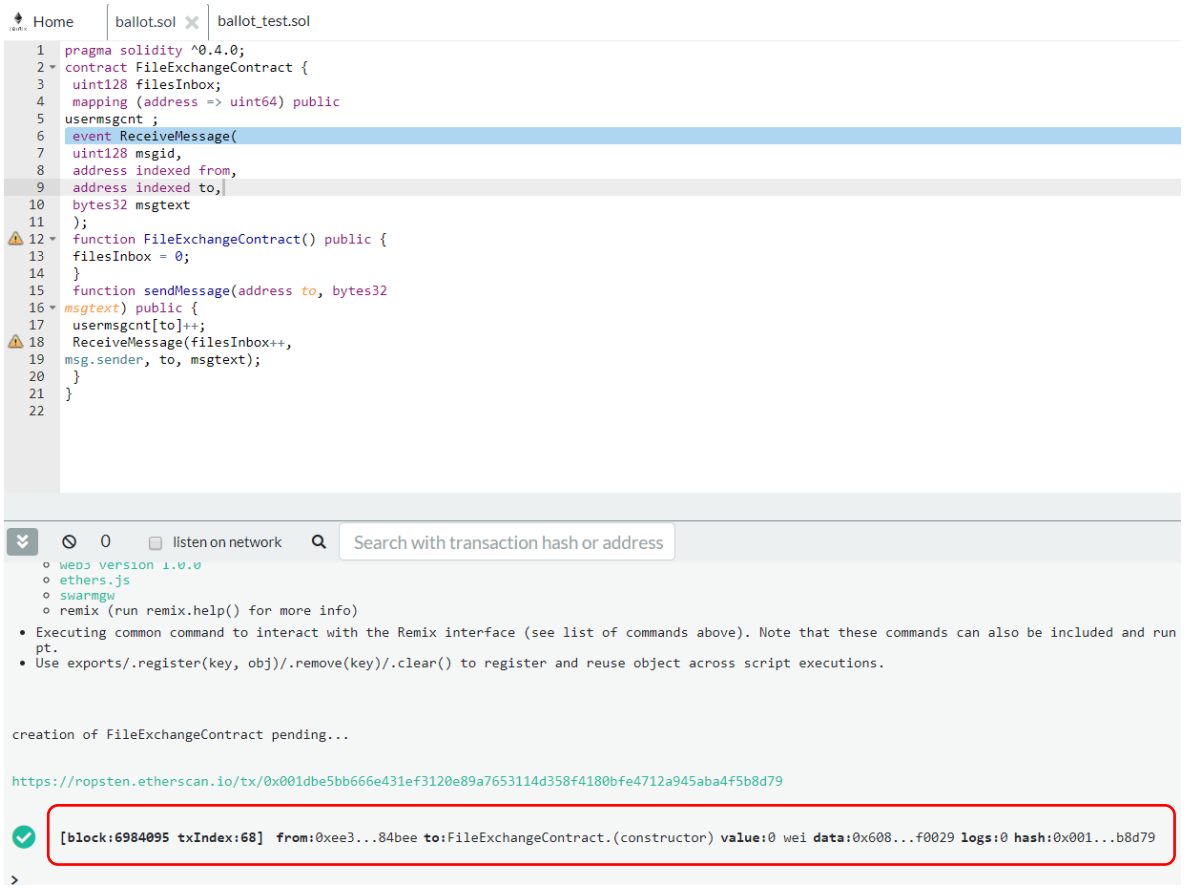



Nos aparecerá la siguiente ventana, y le damos click en confirmar, ya que es el costo de la transacción.

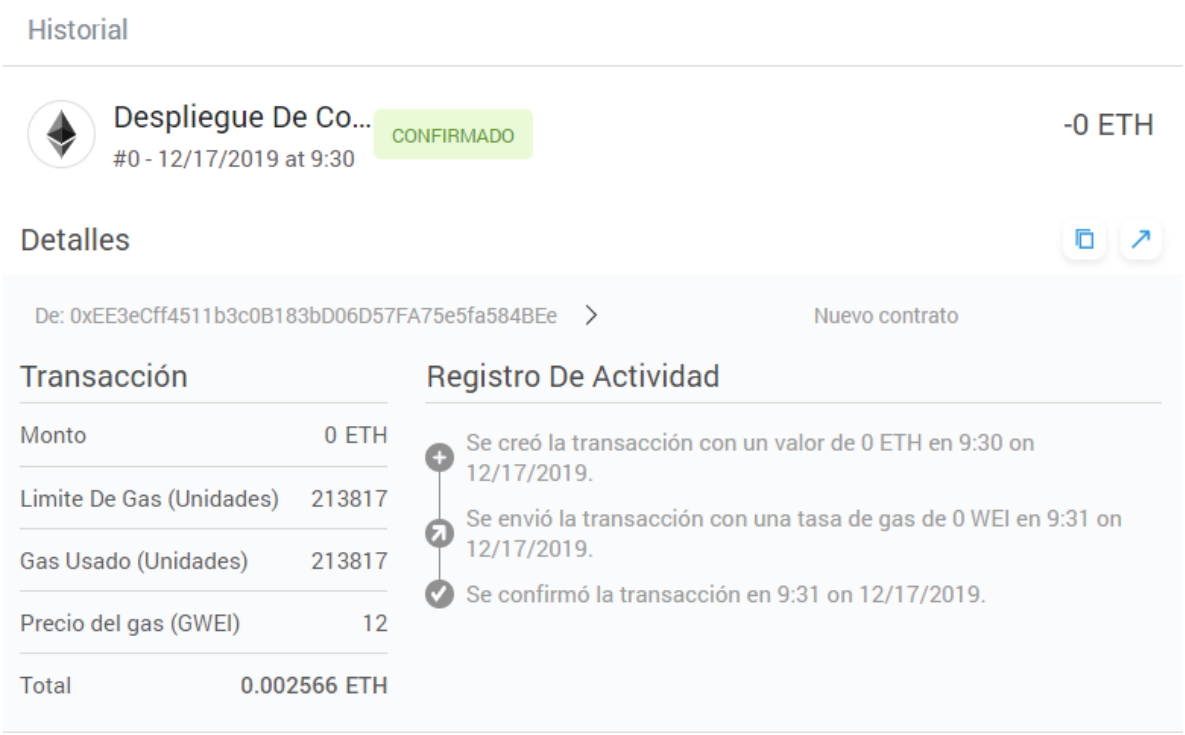


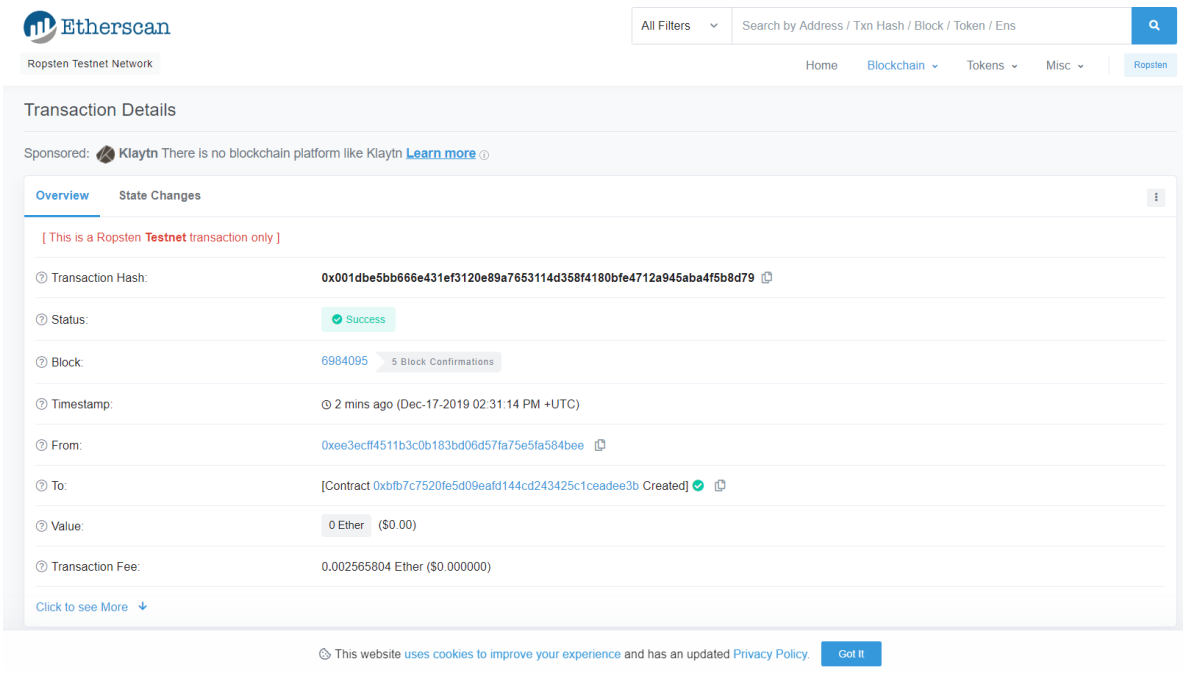

Y si nos vamos a nuestra cuenta de Metamask podremos ver cómo el despliegue del contrato ha sido confirmado, en la parte inferior de REMIX también podremos observar el ID del bloque que identifica la transacción que se acaba de realizar.


METAMASK
Red de prueba Ropsten
Account 1
Detalles
0xEE3e...4BEe
0.9974 ETH
¿No puedes ver tus tokens?
0.9974 ETH
Depósito
Enviar
Historial
Despliegue De Co...
#0 - 12/17/2019 at 9:30
CONFIRMADO
-0 ETH
Depósito
12/17/2019 at 9:23
CONFIRMADO
1 ETH


Home
ballot.sol
ballot_test.sol
pragma solidity ^0.4.0;
contract FileExchangeContract {
uint128 filesInbox;
mapping (address => uint64) public
usermsgcnt ;
event ReceiveMessage(
uint128 msgid,
address indexed from,
address indexed to,
bytes32 msgtext
);
function FileExchangeContract() public {
filesInbox = 0;
}
function sendMessage(address to, bytes32
msgtext) public {
usermsgcnt[to]++;
ReceiveMessage(filesInbox++,
msg.sender, to, msgtext);
}
}
listen on network
Search with transaction hash or address
ethers.js
swarmgw
remix (run remix.help() for more info)
Executing common command to interact with the Remix interface (see list of commands above). Note that these commands can also be included and run
pt.
Use exports/.register(key, obj)/.remove(key)/.clear() to register and reuse object across script executions.
creation of FileExchangeContract pending...
https://ropsten.etherscan.io/tx/0x001dbe5bb666e431ef3120e89a7653114d358f4180bfe4712a945aba4f5b8d79
[block:6984095 txIndex:68] from:0xee3...84bee to:FileExchangeContract.(constructor) value:0 wei data:0x608...f0029 logs:0 hash:0x001...b8d79

Para verlo en EtherScan le damos click al despliegue en Metamask.



## Instalación WEB3

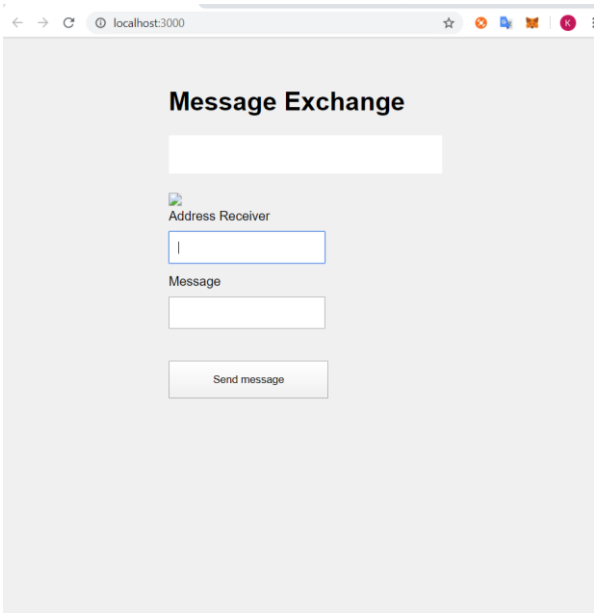
Por último, vamos a crear una carpeta llamada ethereumdapp, que la pueden encontrar en el repositorio. Estando dentro de la carpeta vamos a instalar web3 con el comando: "***npm install web3***".

Tendremos 3 archivos, uno que será el index.html, un archivo para los estilos llamada main.css, y un archivo de configuración llamada web3_connect.js. Todos estarán dentro de la carpeta /ethereumdapp. El archivo de configuración web3_connect.js también lo podemos poner como un script dentro lo que será nuestro index.html.

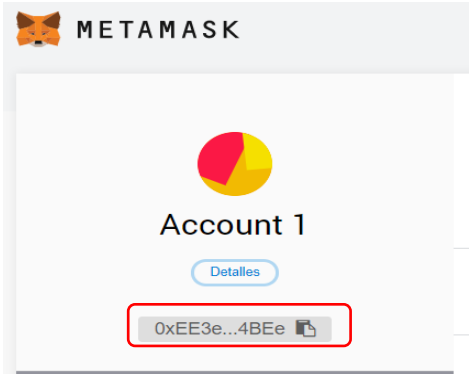
Para correr la aplicación ejecutaremos el siguiente comando: "**npm install lite-server –save-dev**", el cual instalará el web server sobre el cual se desplegará el archivo de configuración. Después de realizar este paso debemos ir al package.json y agregar el script correspondiente, para esto escribiremos:

```
"scripts": {
        "dev": "lite-server"
}
```

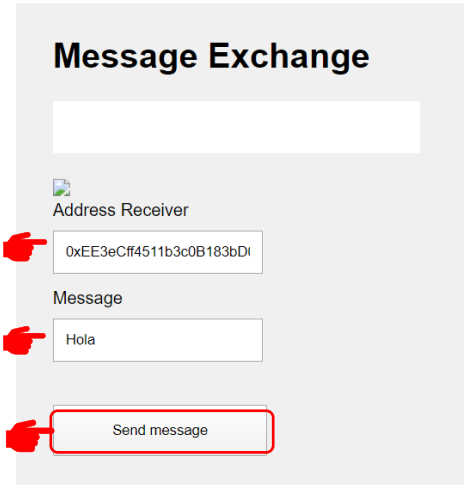
Para iniciar el lite-server, corremos el siguiente comando: "**npm run dev**". Y ahora ingresamos a la dirección http://localhost:3000, nos deberá aparecer una pestaña como esta:




A continuación, en donde dice address receiver pondremos la dirección de nuestra billetera virtual. Para esto puede dirigirse a Metamask y copiar el id de su cuenta.




Ponemos cualquier mensaje, y le damos click al botón de *"send message"*. Sólo queda confirmar la transacción, y listo.




Para entender un poco mejor cada uno de los pasos y para qué sirve cada uno de los archivos ubicados en esta carpeta vaya al siguiente link: https://github.com/jcamilovelandiab/BSIEMv2/blob/master/CREAR%20DAPP%20EN%20ETHEREUM.pdf

# Anexo 2
## Guía de Implementación DAAP de Voto Electrónico

El siguiente tutorial también lo podemos encontrar en el siguiente repositorio: https://github.com/jcamilovelandiab/BSIEMv2/tree/master/election-dapp-master

Vamos a construir una dapp básica de voto electrónico. Vamos a usar las siguientes aplicaciones:

- Ganache: Esta aplicación ofrece blockchain privada de Ethereum que podemos usar para ejecutar pruebas, ejecutar comandos e inspeccionar el estado de la blockchain.
- Metamask: Esta es una extensión para acceder a las aplicaciones distribuidas habilitadas por Ethereum.
- Remix: Es un IDE de Ethereum en línea. Vamos a usar esto para probar y desplegar nuestros contratos.

### Contrato del voto electrónico

Este es el contrato que vamos a desplegar en la blockchain. Este es un contrato simple, y como podemos ver nos permite escribir el nombre de nuestro candidato.

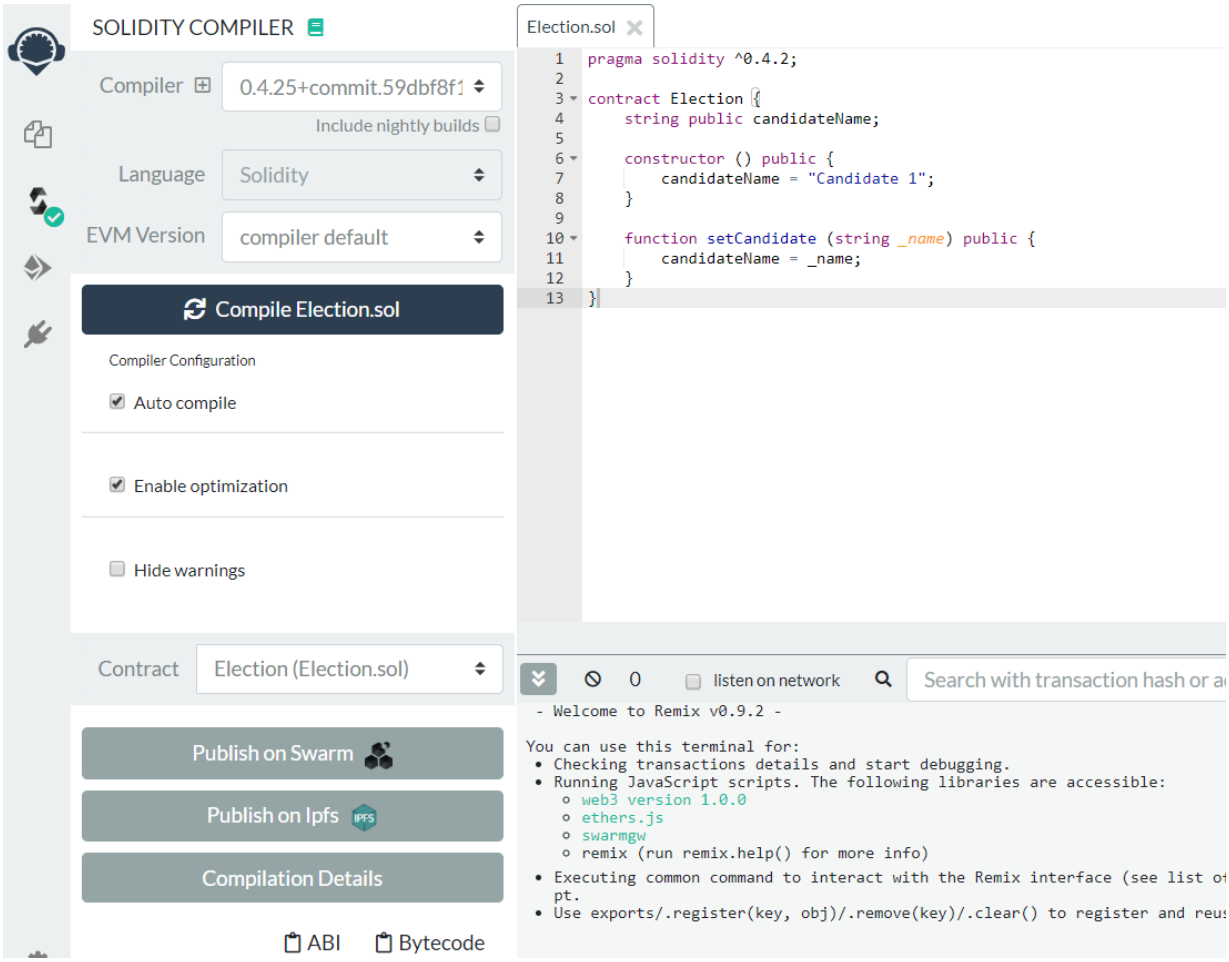
```
pragma solidity ^0.4.2;

contract Election {
    string public candidateName;

    constructor () public {
        candidateName = "Candidate 1";
    }

    function setCandidate (string _name) public {
        candidateName = _name;
    }
}
```

El código del contrato también lo podemos encontrar el siguiente link https://github.com/jcamilovelandiab/BSIEMv2/blob/master/election-dapp-master/contracts/Election.sol

Abrimos el IDE de Remix http://remix.ethereum.org/, y activamos los siguientes plugins en configuración:

- Solidity Compiler.
- Deploy & Run transactions.

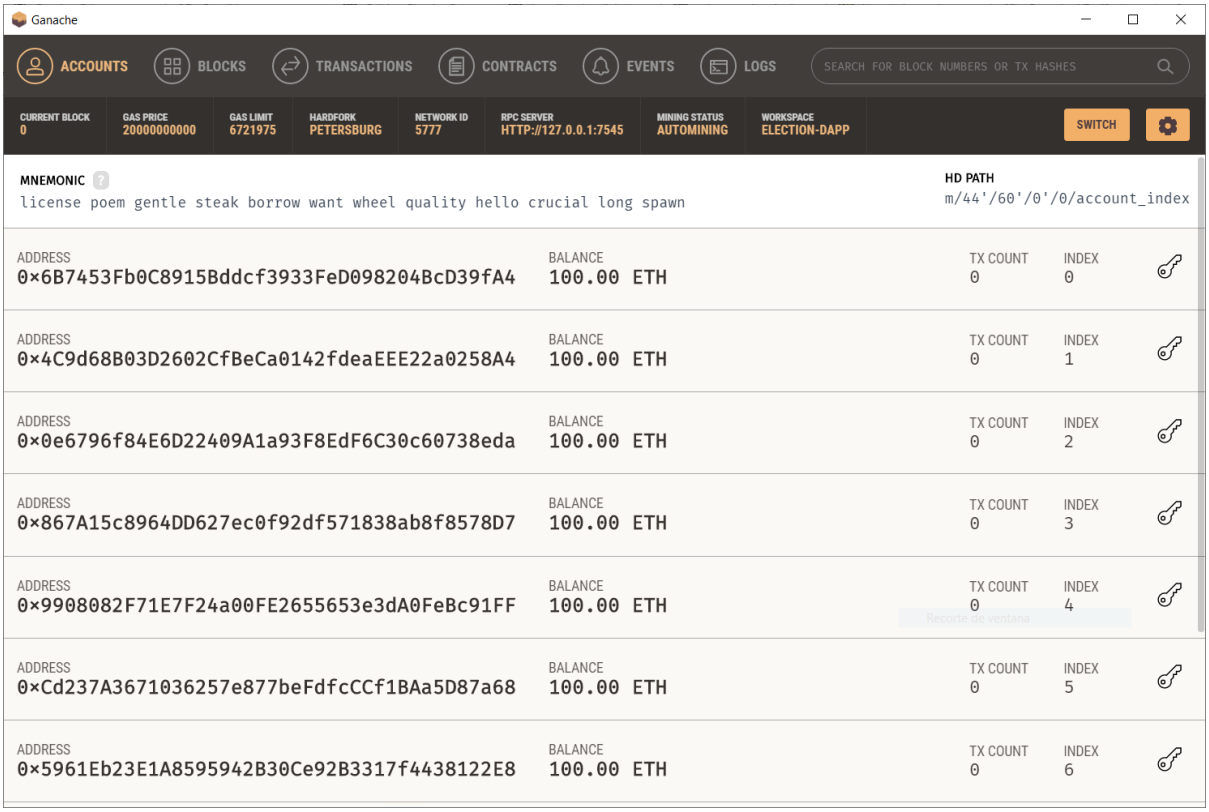
Después de la activación de los plugins, vamos a la pestaña Solidity Compiler, y seleccionamos la versión del compilador 0.4.25+commit.59dbf8f1, y luego creamos el archivo Election.sol y lo compilamos.




## Creando una red blockchain en Ganache.

Podemos descargar Ganache en https://www.trufflesuite.com/ganache. Luego de descargar ganache, abrimos la aplicación y creamos un nuevo espacio de trabajo llamado "election-dapp".

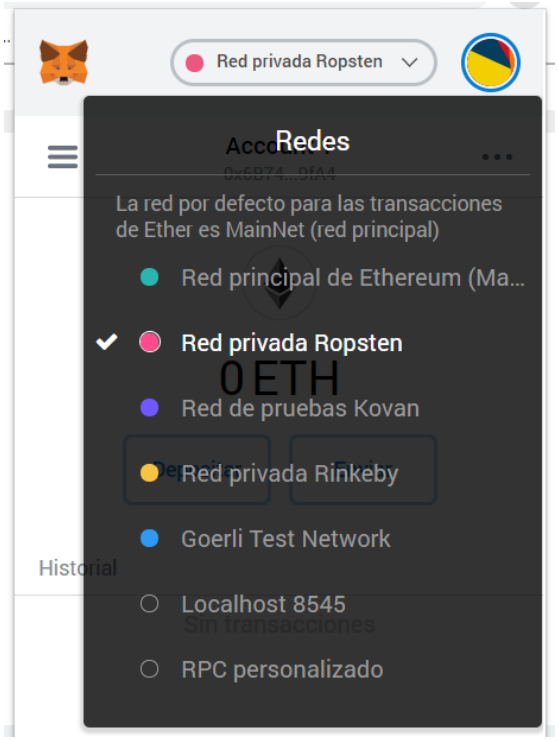
Después de crear el proyecto en Ganache, vemos que Ganache creó 10 billeteras virtuales. También vemos la palabra MNEMONIC. Podemos ver la actividad de las billeteras en MetaMask a través de esta palabra especial que Ganache creó para nosotros.

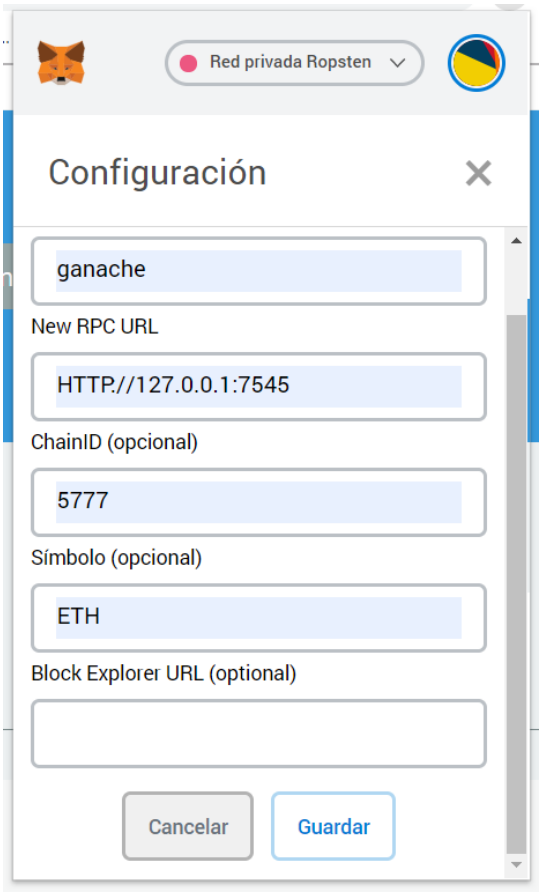



**Conectando Ganache con MetaMask.**

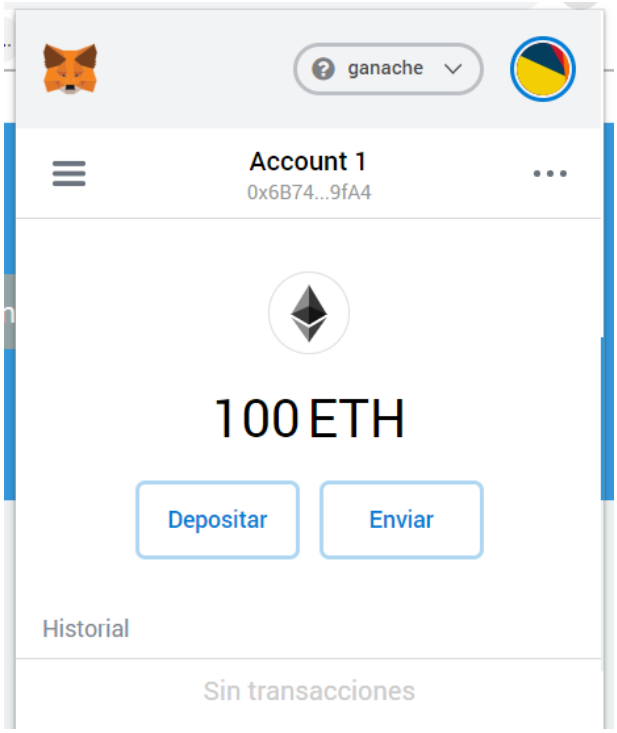
Añadimos la extensión Metamask a Google Chrome. A continuación, introducimos la semilla de la billetera (palabra MNEMONIC) y creamos la contraseña de la cuenta.

Luego, creamos una nueva red en MetaMask para Ganache.



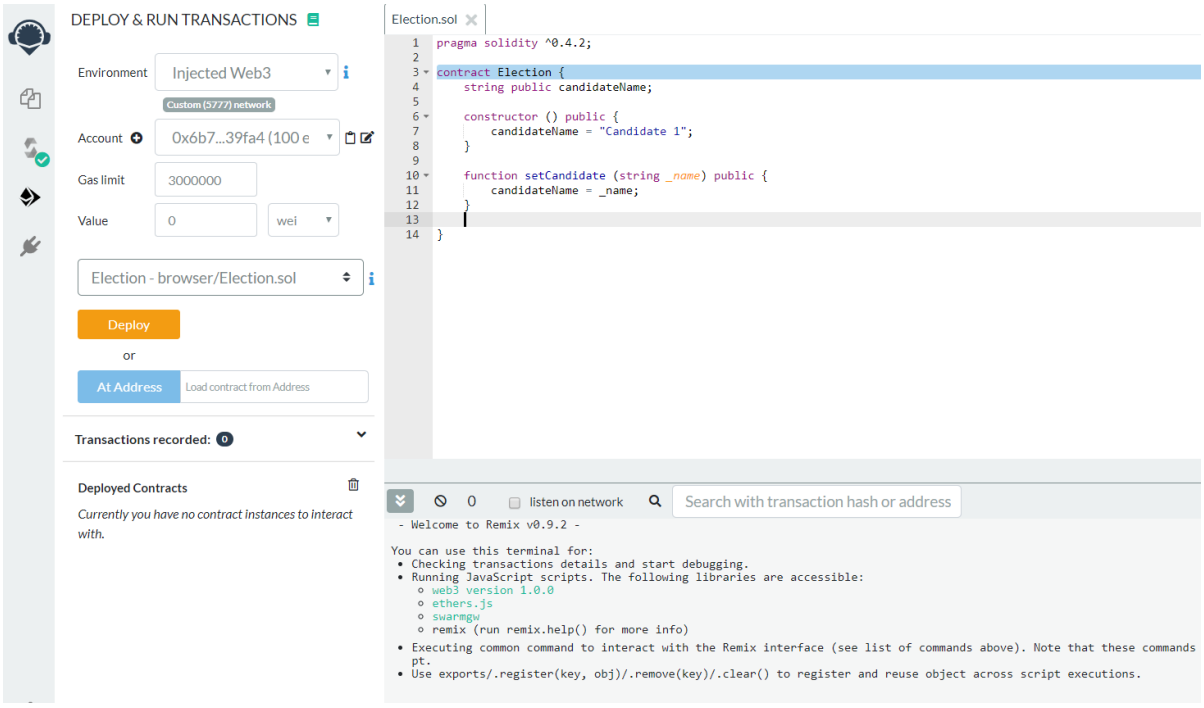



Finalmente, vemos que pudimos acceder a la billetera. Como podemos ver, es la misma billetera con 100ETH que Ganache creó para nosotros.

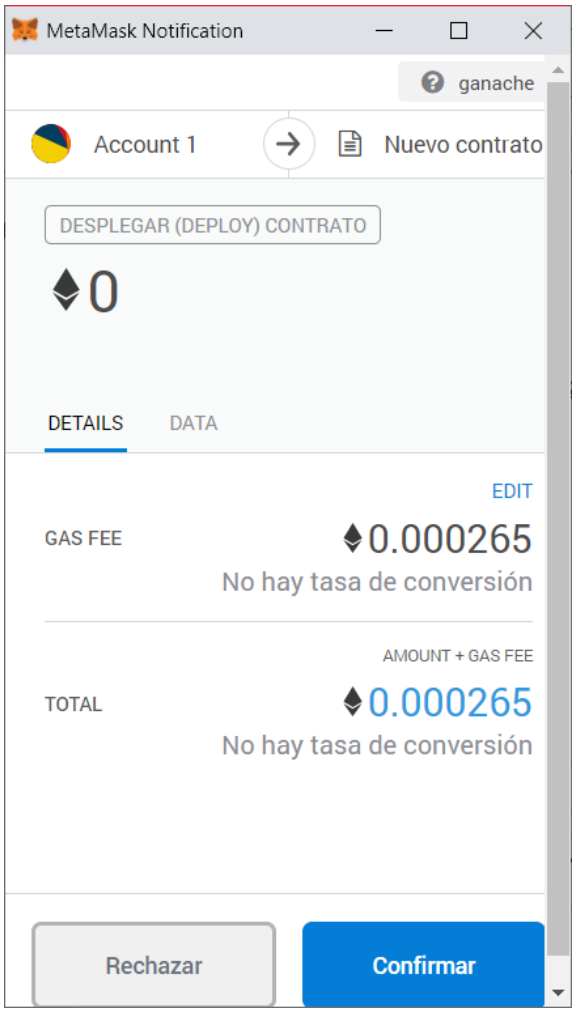



**Desplegando el contrato Smart con Remix.**

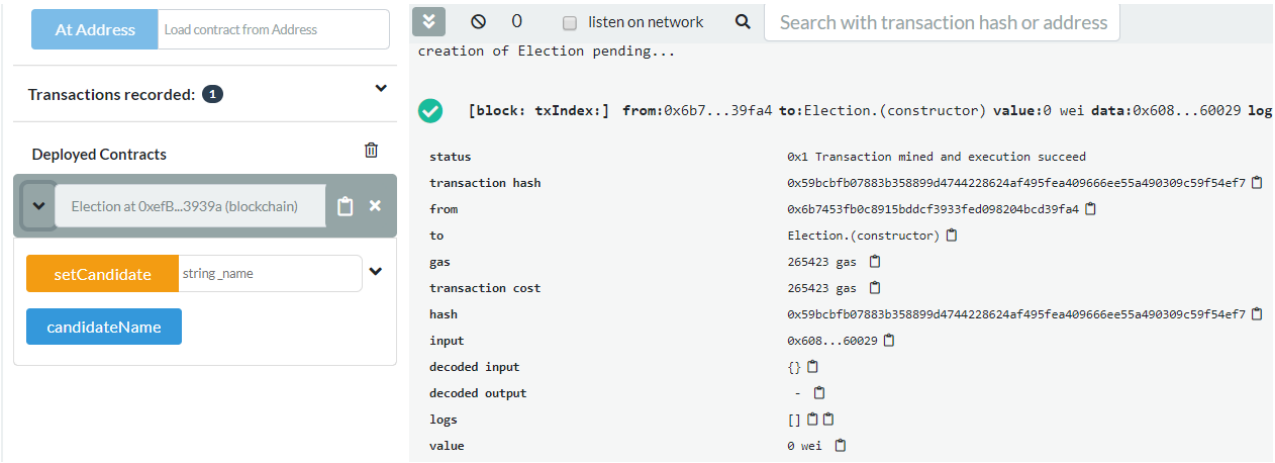
Después de que MetaMask y Remix se enlazan (se enlazan entre sí por sí mismos), seleccionamos el entorno de Injection Web3 y desplegamos el contrato de elección.



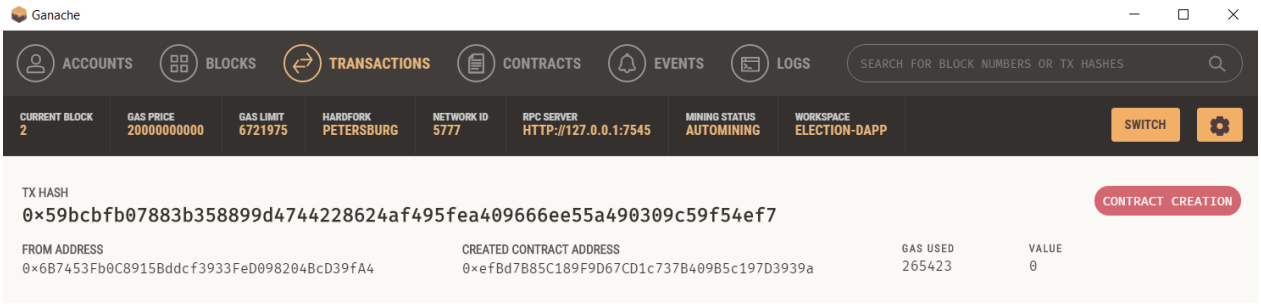
MetaMask muestra el costo del despliegue del contrato, y luego confirmamos la transacción.



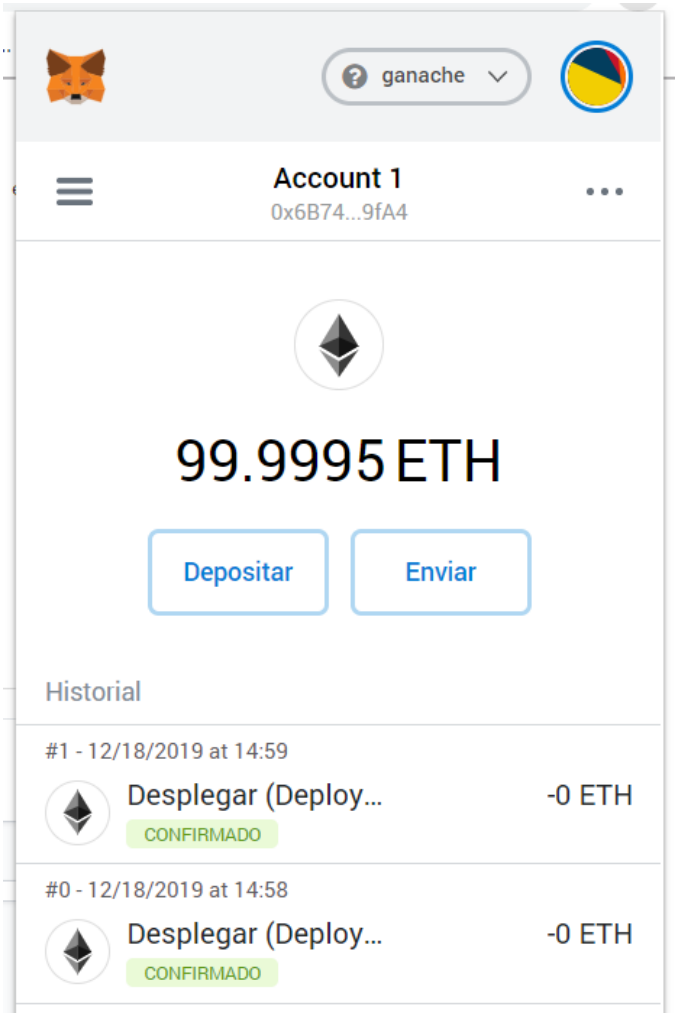

Y, vemos que el contrato fue desplegado con éxito.



En ganache vemos algo que la transacción fue ejecutada, y fue una creación de contrato.




En Metamask vemos que la transacción fue ejecutada, y la billetera tiene menos ETH.



### Vinculando el sitio web con la blockchain

Clonamos este repositorio con el comando:

git clone https://github.com/jcamilovelandiab/ElectionDAPP"

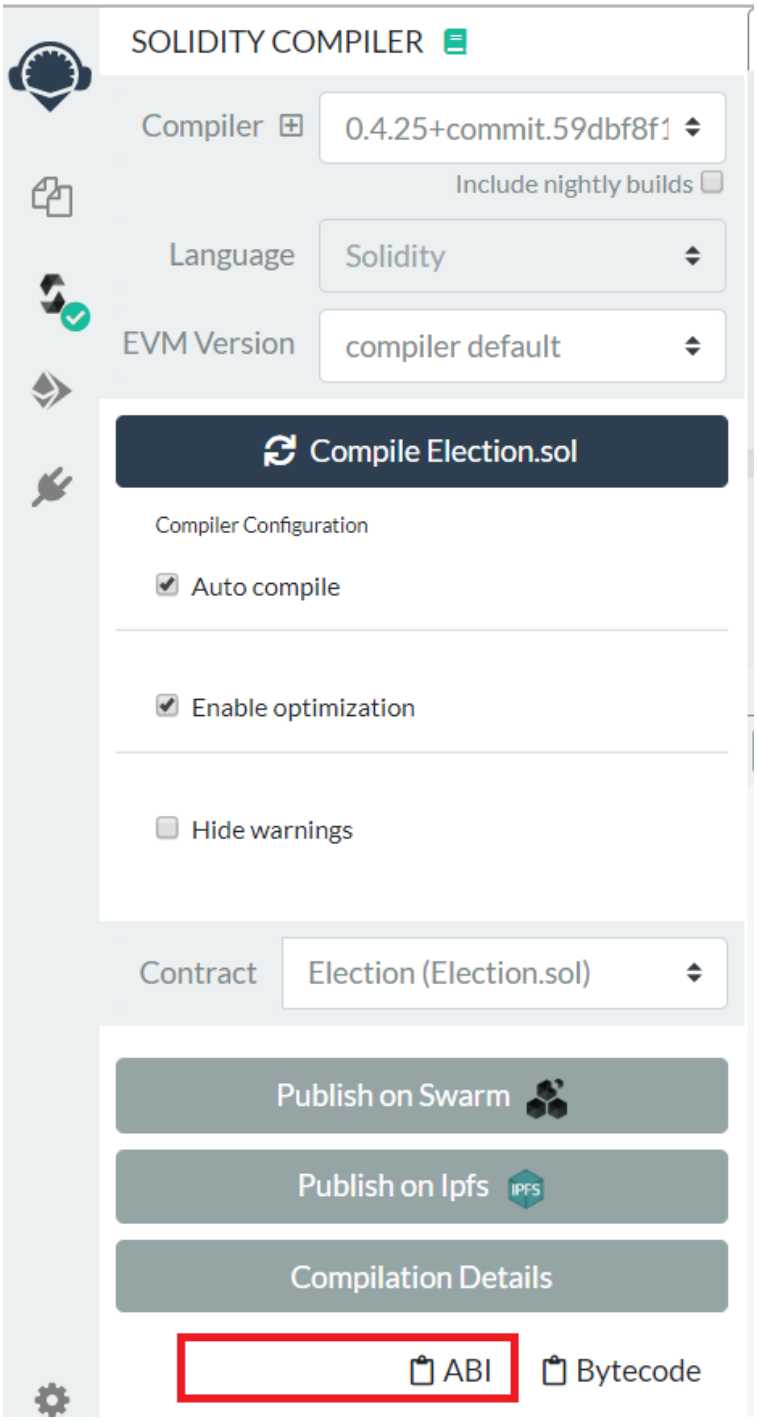
Podemos ver en index.html este script.

index.html

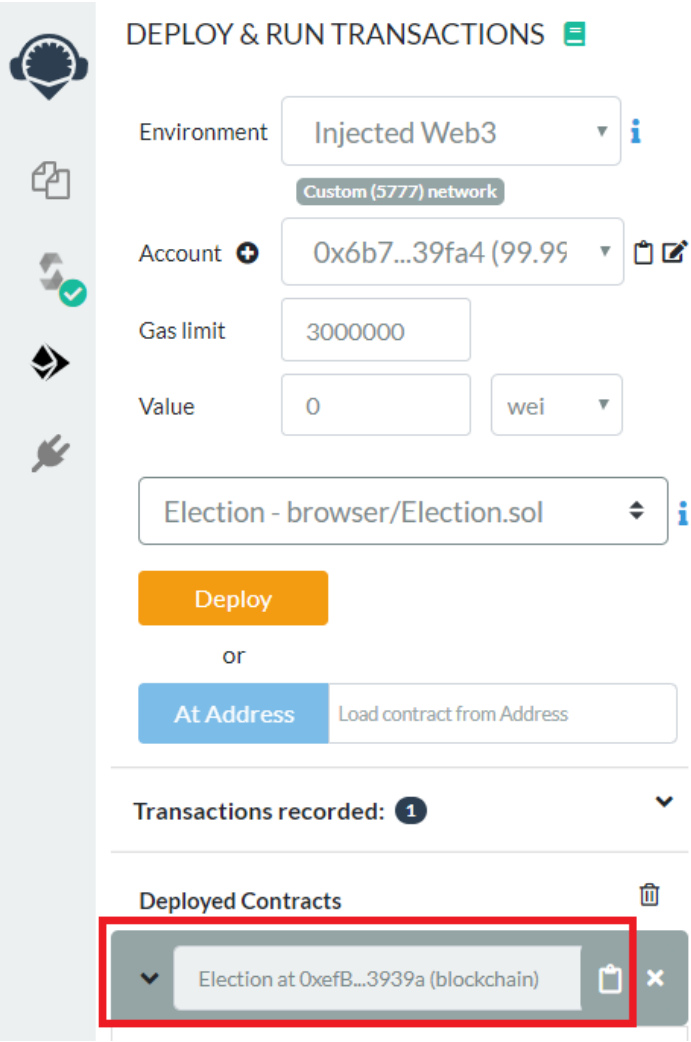
index.html > html > body > script

```
<script src="js/bootstrap.min.js"></script>
<script src="js/web3.min.js"></script>
<script>
  // Initialize Web3
  if (typeof web3 !== 'undefined') {
    web3 = new Web3(web3.currentProvider);
  } else {
    web3 = new Web3(new Web3.providers.HttpProvider('http://127.0.0.1:7545'));
  }
  console.log(web3.eth.accounts);

  // Set Account
  web3.eth.defaultAccount = web3.eth.accounts[0];
  // Set Contract Abi
  var contractAbi = []; // Add Your Contract ABI here!!!
  console.log("ABI->\n"+contractAbi);
  // Set Contract Address
  var contractAddress = ''; // Add Your Contract address here!!!

  // Set the Contract
  var contract = web3.eth.contract(contractAbi).at(contractAddress);
  console.log("contract->"+contract);

  // Display Candidate Name
  contract.candidateName(function(err, candidateName) {
    $('#candidateName').html(candidateName);
  });

  // Change the Candidate Name
  $('form').on('submit', function(event) {
    event.preventDefault();
    window.location.reload();
    contract.setCandidate($('input').val());
  });

</script>
</body>
</html>
```

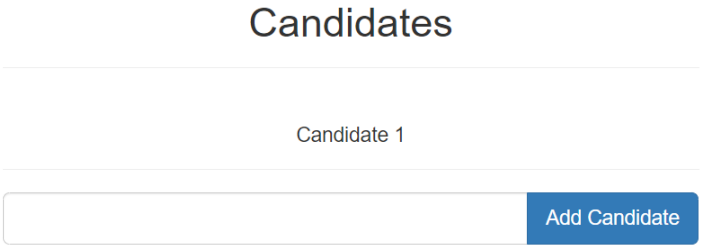
Como podemos ver, la web3 se está conectando a http://127.0.0.1/7545 que es la url de la red Ganache. En la variable contractAbi ponemos el contrato electoral ABI. Podemos obtenerlo de Remix.



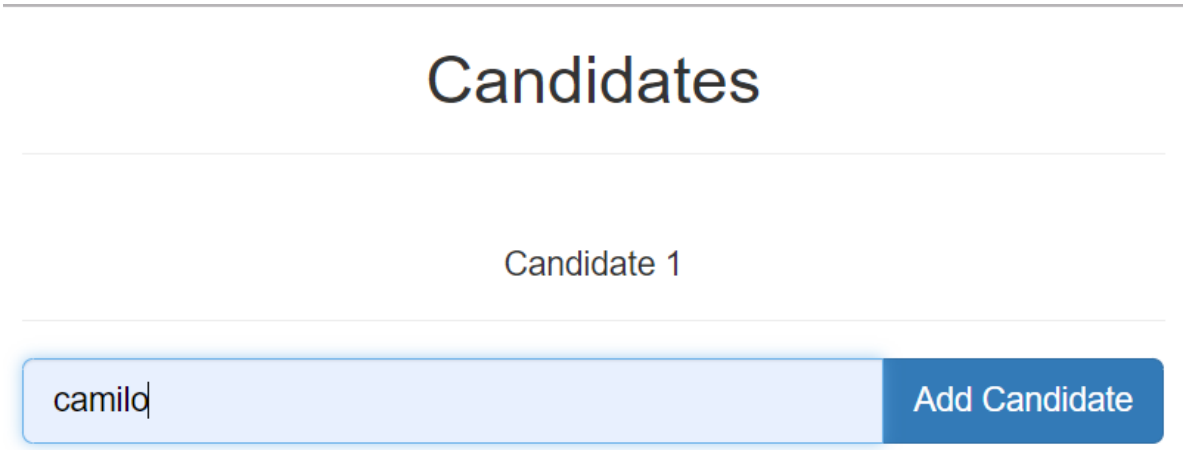

Y, en la variable contractAddress ponemos la dirección que podemos obtener de Remix también.



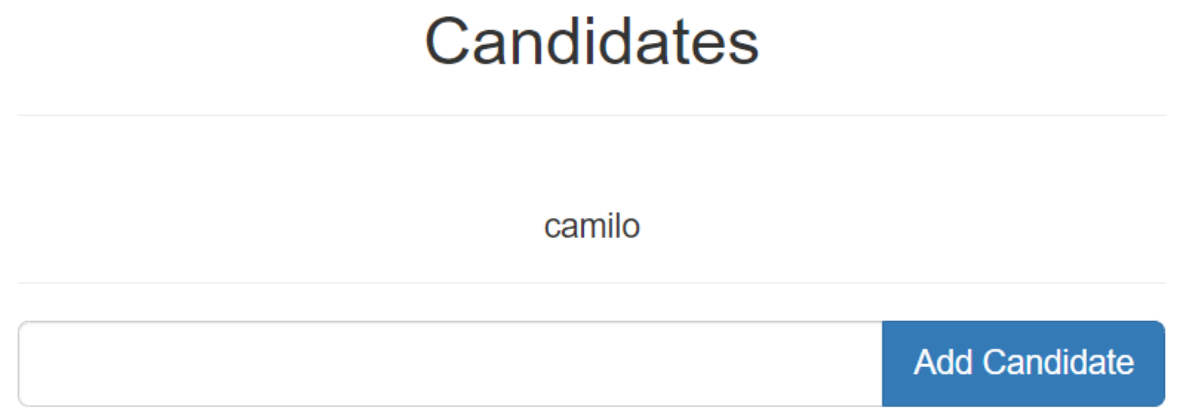

Entonces, sólo abrimos el index.html, y probamos que la aplicación se comunique con la blockchain, y vemos que funciona correctamente.



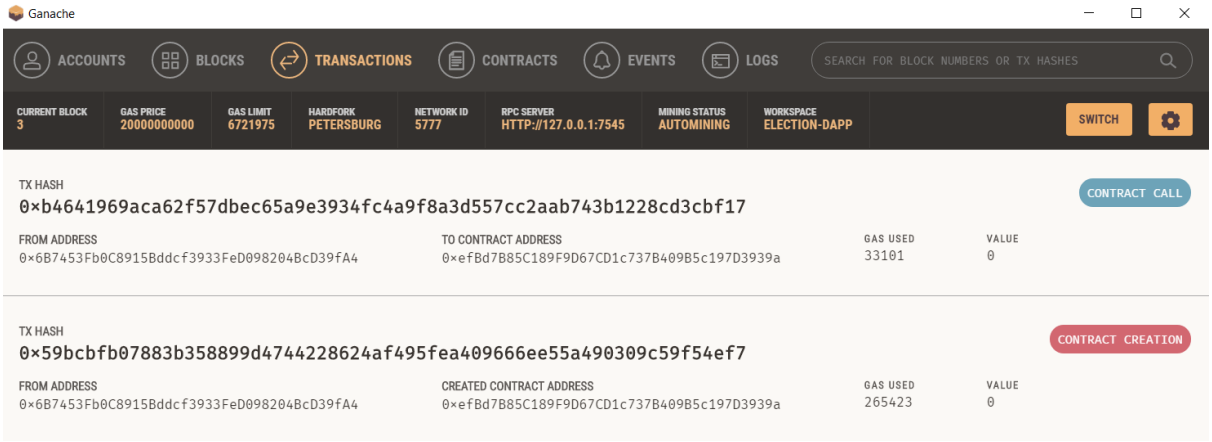
Ingresamos el nombre del candidato.




Vemos que el nombre del candidato ha cambiado.




Comprobamos y vemos en Ganache que se ha creado un bloque y se ha añadido a la blockchain, también que la transacción que se hizo fue desde una llamada al contrato.




¡Acabamos!, ya tenemos nuestra página web electoral que funciona con una blockchain.