

OPEN SOURCE BLOCKCHAIN

Desarrollo de Tecnologías Emergentes – TG3



7 de mayo de 2018

GRUPO T1

Contenido

[1. Autores del trabajo, planificación y entrega 3](#_Toc513290076)

[1.1 Autores 3](#_Toc513290077)

[1.2 Planificación 3](#_Toc513290078)

[1.3 Entrega 3](#_Toc513290079)

[2. Requisitos del prototipo a implementar 4](#_Toc513290080)

[2.1 Requisitos funcionales 4](#_Toc513290081)

[2.2 Requisitos NO funcionales 5](#_Toc513290082)

[3. Criterios de comparación en la implementación 6](#_Toc513290083)

[3.1 Criterio 1: Documentación 6](#_Toc513290084)

[3.2 Criterio 2: Horas empleadas 6](#_Toc513290085)

[3.3 Criterio 3: Líneas de código 6](#_Toc513290086)

[3.4 Criterio 4: Facilidad de implementación 6](#_Toc513290087)

[3.5 Criterio 5: Velocidad de arranque 6](#_Toc513290088)

[3.6 Criterio 6: Velocidad de funcionamiento 7](#_Toc513290089)

[3.7 Criterio 7: Cantidad de nodos 7](#_Toc513290090)

[3.8 Criterio 8: Caída de los nodos 7](#_Toc513290091)

[3.9 Criterio 9: Soporte 7](#_Toc513290092)

[4. Proyecto de implementación de un prototipo del sistema utilizando MultiChain 8](#_Toc513290093)

[4.1 Documentación de diseño 8](#_Toc513290094)

[4.2 Documentación de construcción 8](#_Toc513290095)

[4.3 Documentación de pruebas 14](#_Toc513290096)

[4.3.1 Comprobaciones 14](#_Toc513290097)

[4.3.2 Errores 15](#_Toc513290098)

[4.4 Documentación de instalación 17](#_Toc513290099)

[4.4.1 Proceso de instalación de las máquinas virtuales 17](#_Toc513290100)

[4.4.2 Instalación 19](#_Toc513290101)

[4.5 Manual de usuario 23](#_Toc513290102)

[5. Proyecto de implementación de un prototipo del sistema utilizando la Monax 24](#_Toc513290103)

[5.1 Documentación de diseño 24](#_Toc513290104)

[5.2 Documentación de construcción 24](#_Toc513290105)

[5.3 Documentación de pruebas 26](#_Toc513290106)

[5.4 Documentación de instalación 28](#_Toc513290107)

[5.4.1 Proceso de instalación de las máquinas virtuales 28](#_Toc513290108)

[5.4.2 Instalación de Docker y Monax 30](#_Toc513290109)

[5.5 Manual de usuario 32](#_Toc513290110)

[6. Comparación de las dos implementaciones 33](#_Toc513290111)

[6.1 Evaluación de los criterios en la implementación usando MultiChain 33](#_Toc513290112)

[6.2 Evaluación de los criterios en la implementación usando Monax 33](#_Toc513290113)

[7. Comparación de la implementación de las tecnologías 34](#_Toc513290114)

[8. Conclusiones 36](#_Toc513290115)

[ANEXO 1: Otros Requisitos 36](#_Toc513290116)

[A.1. MultiChain 36](#_Toc513290117)

[A.1.1. Requisitos Funcionales 36](#_Toc513290118)

[A.1.2. Requisitos NO Funcionales 36](#_Toc513290119)

[A.2 Monax 36](#_Toc513290120)

[A.2.1. Requisitos Funcionales 36](#_Toc513290121)

[A.2.2. Requisitos NO Funcionales 36](#_Toc513290122)

# 1. Autores del trabajo, planificación y entrega

## 1.1 Autores

Este grupo es el T1 y está formado por:

* Alejandro Díaz Moreno (Coordinador del grupo T1)
* Daniel Ortega Expósito
* Daniela Guzmán Pisfil
* María Castro Vaquerizo
* Paula Hernández Jordá

## 1.2 Planificación

Como la herramienta GanttPro ya no permite compartir mediante URL la planificación hemos tenido que añadir al profesor de la asignatura a la misma.

URL 🡪[https://app.ganttpro.com/#!/app/home](https://app.ganttpro.com/%23!/app/home)

## 1.3 Entrega

Indicamos el enlace (URL) a un repositorio en GitHub:

<https://github.com/alex7dm/DTE_T1_OpenSourceBlockchain>

En dicho repositorio encontraremos:

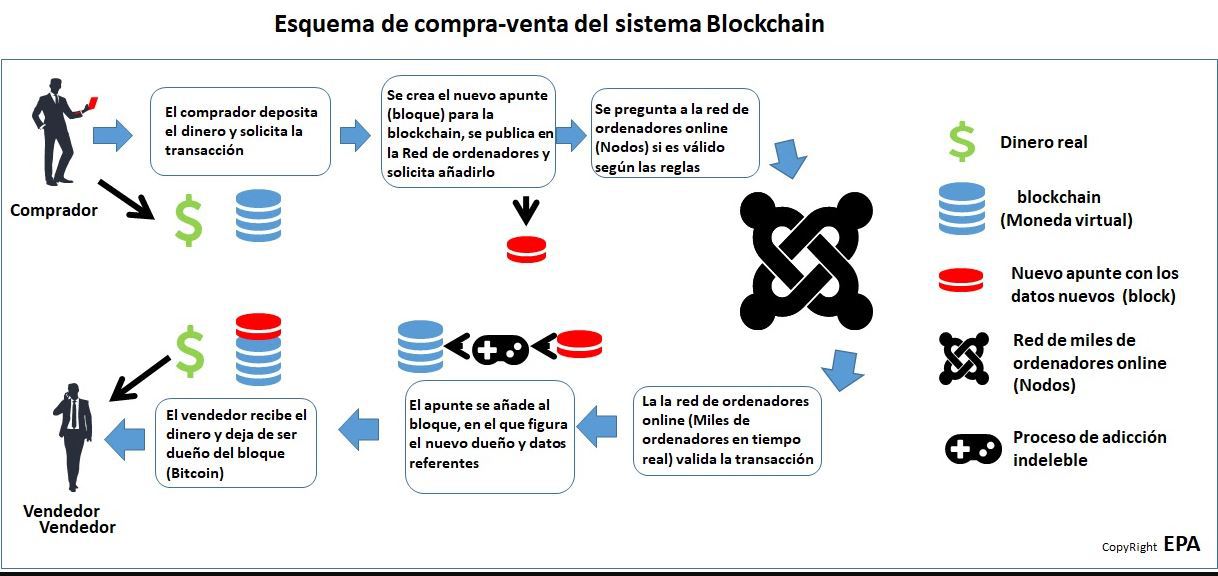
* El trabajo terminado con el nombre **TG3\_final.pdf**
* La presentación del trabajo con el nombre **TG3\_final.pptx**
* Prototipos obtenidos implementando con las tecnologías escogidas:
  + **MultiChain\_final.zip**
  + **Monax\_final.zip**

# 2. Requisitos del prototipo a implementar

Vamos a utilizar BlockChain para realizar una transacción que permita realizar un intercambio de activos entre dos o más partes.

Por ejemplo, el envío de un activo (dólares) de Marta a Jorge, al mismo tiempo que envía un activo en euros de Bob a Alicia. Como el intercambio se realiza con una sola transacción significa que, o todas las transacciones de activos tienen lugar simultáneamente o ninguna tiene lugar.

En el mundo de las finanzas este tipo de transacciones se denomina entrega contra pago (DvP)



A continuación, se detallan todos los requisitos para implementar:

## 2.1 Requisitos funcionales

Los requisitos funcionales deben ser los mismos para las dos implementaciones.

En la siguiente tabla se indicará el catálogo de requisitos funcionales del sistema.

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
| RF01 | Realizar la transacción |
| RF02 | Instalar la BlockChain |
| RF03 | Conectarse a la BlockChain creada |
| RF04 | Instalación de los entornos |
| RF05 | Cada transacción tendrá un numero identificativo diferente |

## 2.2 Requisitos NO funcionales

En la siguiente tabla se indicará el catálogo de requisitos no funcionales del sistema.

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
| R01 | Documentación |
| R02 | El retardo de la transacción no debe ser excesivo |
| R03 | Seguridad |
| R04 | Implementación en Linux |
| R05 | La implementación se debe poder llevar a cabo en una máquina virtual sencilla |
| R06 | La implementación se debe poder llevar a cabo en una máquina con 2 GB de RAM |
| R07 | La implementación se debe poder llevar a cabo en una CPU con 100% de límite de ejecución |
| R08 | Se utilizará la versión 5.2.4 de Virtual Box |
| R09 | Después de la implementación de la BlockChain, el número medio de errores cometidos por desarrollador no excederá de dos por día. |

# 3. Criterios de comparación en la implementación

En el trabajo TG2 se definieron criterios de comparación de las dos tecnologías a nivel teórico.

En este trabajo hay que definir criterios para la comparación de la implementación de las tecnologías en la construcción del prototipo de sistema de ejemplo, cuyos requisitos son los establecidos en el apartado 2.

## 3.1 Criterio 1: Documentación

**Nombre del criterio:** Documentación.

**Descripción:** Documentación encontrada para el desarrollo de la actividad.

**Tipo de valor:** Numérico

## 3.2 Criterio 2: Horas empleadas

**Nombre del criterio:** Horas empleadas en el desarrollo del sistema.

**Descripción:** Horas invertidas en el desarrollo de la tecnología

**Tipo de valor:** Numérico (horas).

## 3.3 Criterio 3: Líneas de código

**Nombre del criterio:** Líneas de código.

**Descripción:** Número de líneas programadas.

**Tipo de valor:** Numérico (líneas).

## 3.4 Criterio 4: Facilidad de implementación

**Nombre del criterio:** Facilidad de implementación

**Descripción:** Facilidad de reacción frente a la implementación o los fallos producidos.

**Tipo de valor:** Booleano (Si/No)

## 3.5 Criterio 5: Velocidad de arranque

**Nombre del criterio:** Velocidad de funcionamiento

**Descripción:** Tiempo que tarda en iniciarse el sistema y conectarse lo nodos.

**Tipo de valor:** Texto libre

## 3.6 Criterio 6: Velocidad de funcionamiento

**Nombre del criterio:** Velocidad de funcionamiento

**Descripción:** Tiempo que tarda en mostrar respuestas por pantalla.

**Tipo de valor:** Texto libre

## 3.7 Criterio 7: Cantidad de nodos

**Nombre del criterio:** Cantidad de nodos

**Descripción:** Número de nodos necesarios para el funcionamiento de la BlockChain.

**Tipo de valor:** Numérico.

## 3.8 Criterio 8: Caída de los nodos

**Nombre del criterio:** Caída de los nodos.

**Descripción:** Número de veces que se han caído los nodos desde su creación.

**Tipo de valor:** Numérico.

## 3.9 Criterio 9: Soporte

**Nombre del criterio:** Soporte.

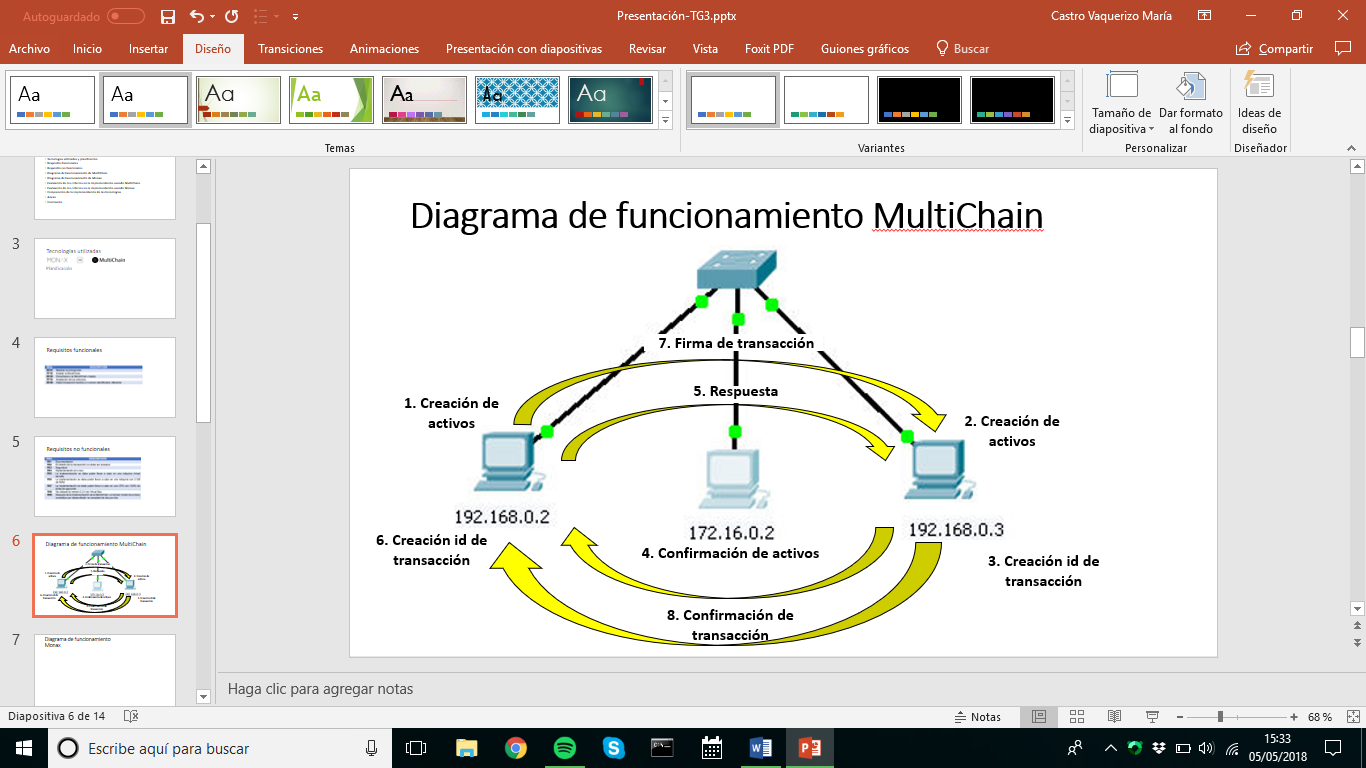
**Descripción:** Nivel de calidad de respuesta antes las dudas planteadas en su página oficial

**Tipo de valor:** Escala (1-5)

# 4. Proyecto de implementación de un prototipo del sistema utilizando MultiChain

## 4.1 Documentación de diseño

Diagrama de funcionamiento de MultiChain

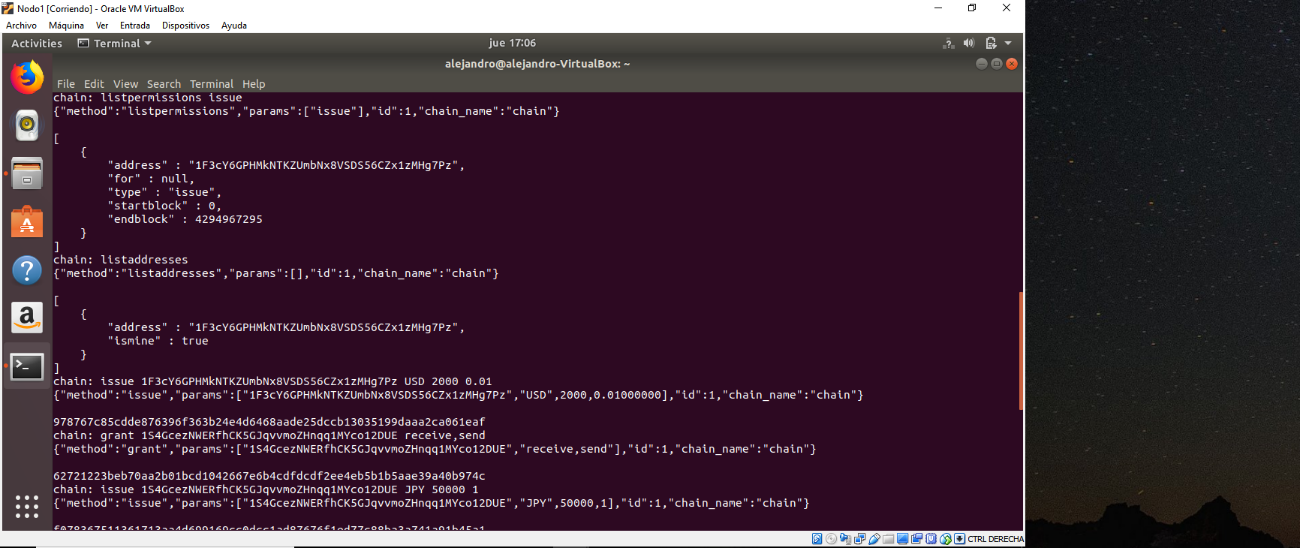


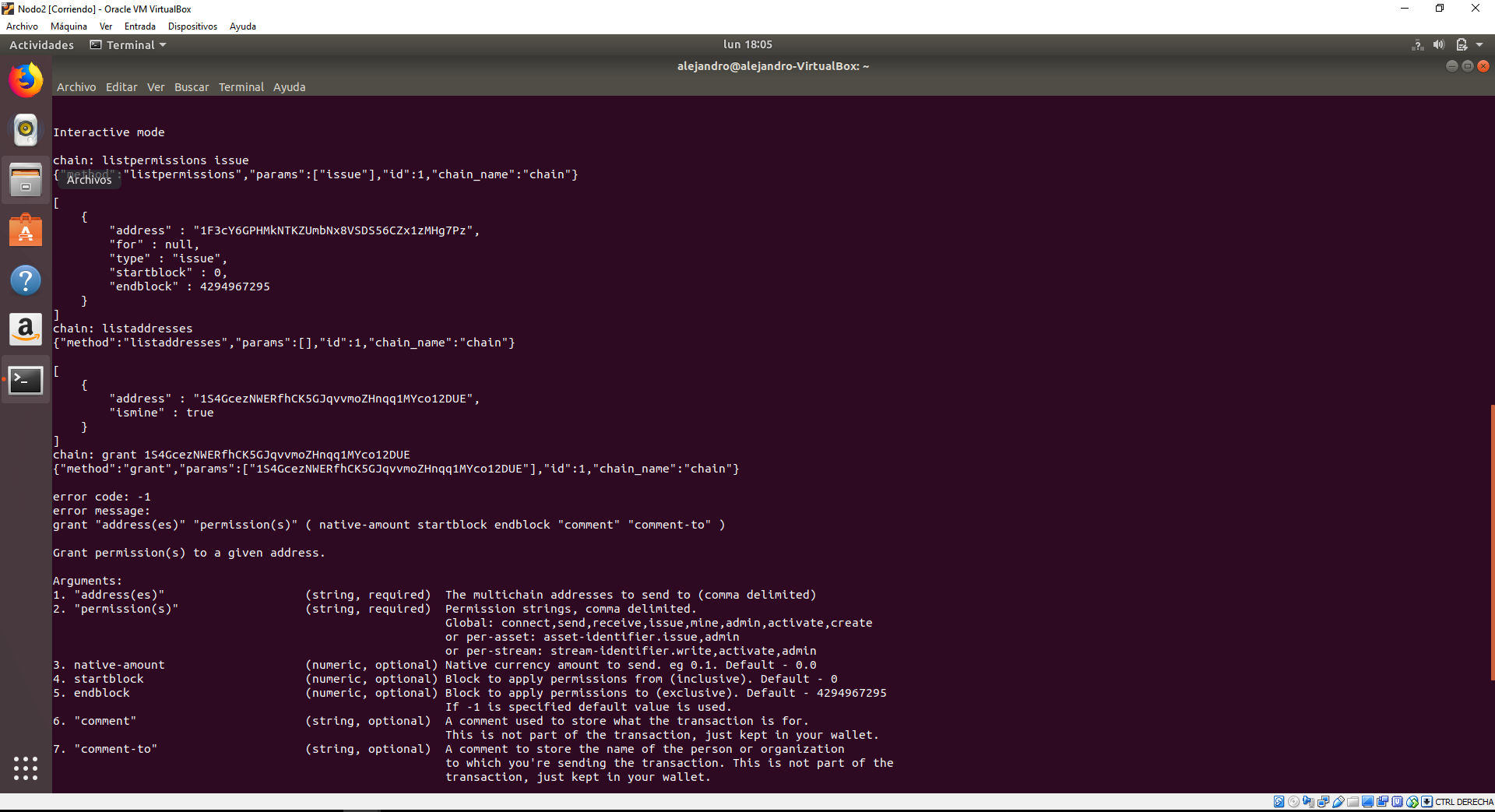
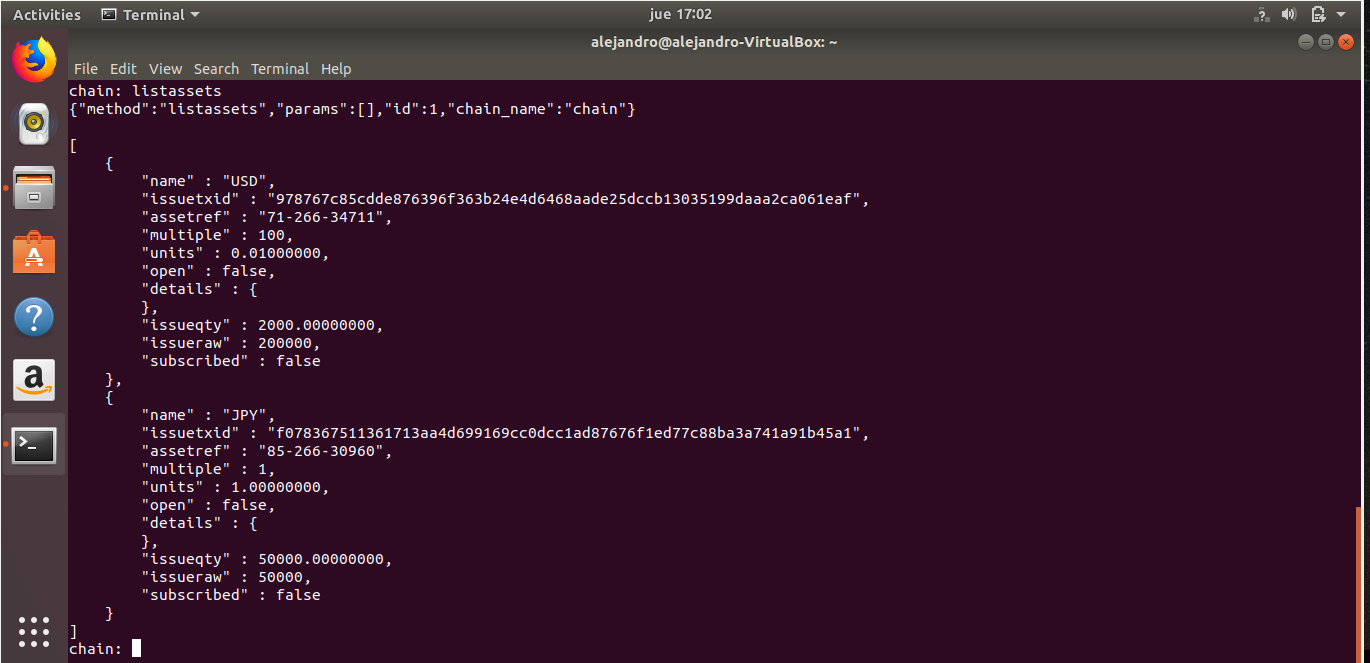
## 4.2 Documentación de construcción

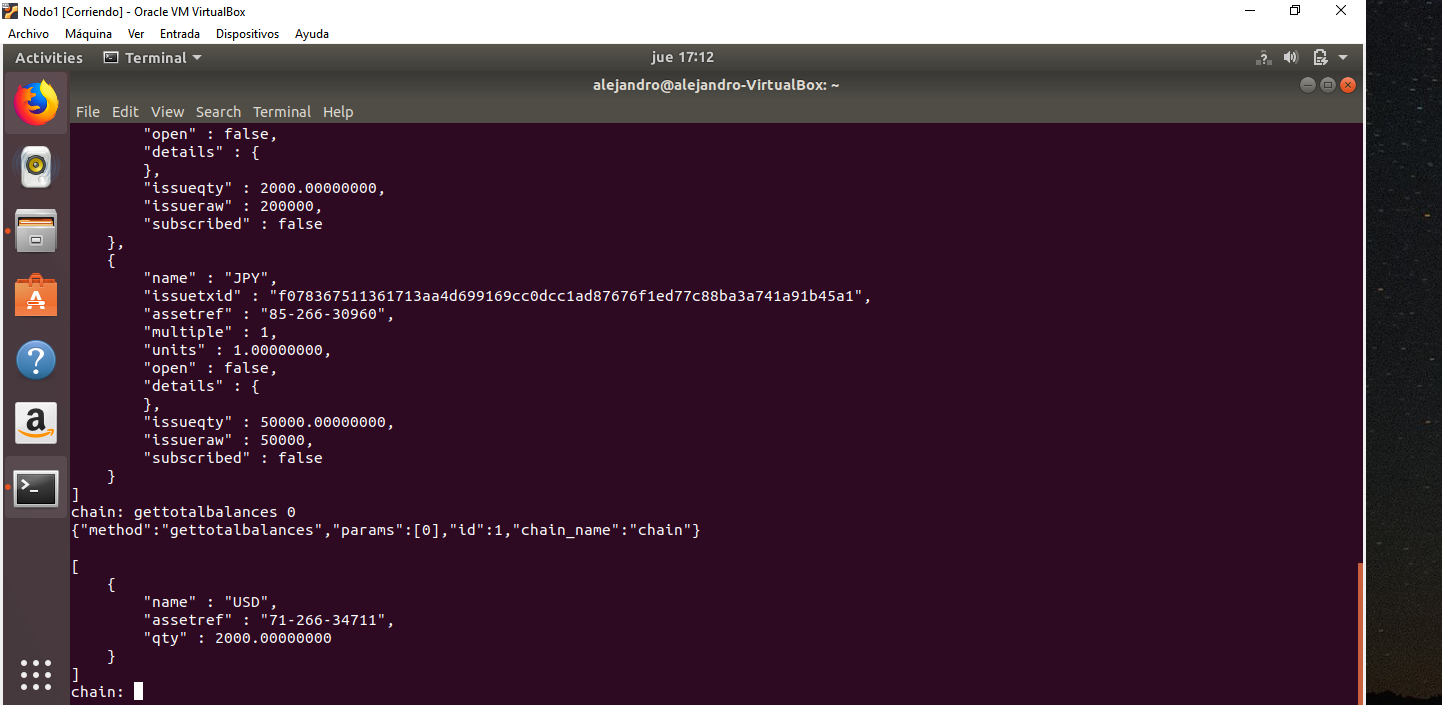
A continuación, explicamos paso a paso como construimos y realizamos la transacción.

Primero hay que entrar en la chain en modo interactivo en ambos nodos usando el comando multichain-cli {nombre de la chain}

Listar la dirección que tiene permiso para crear activos. Se crea el activo en el nodo 1.

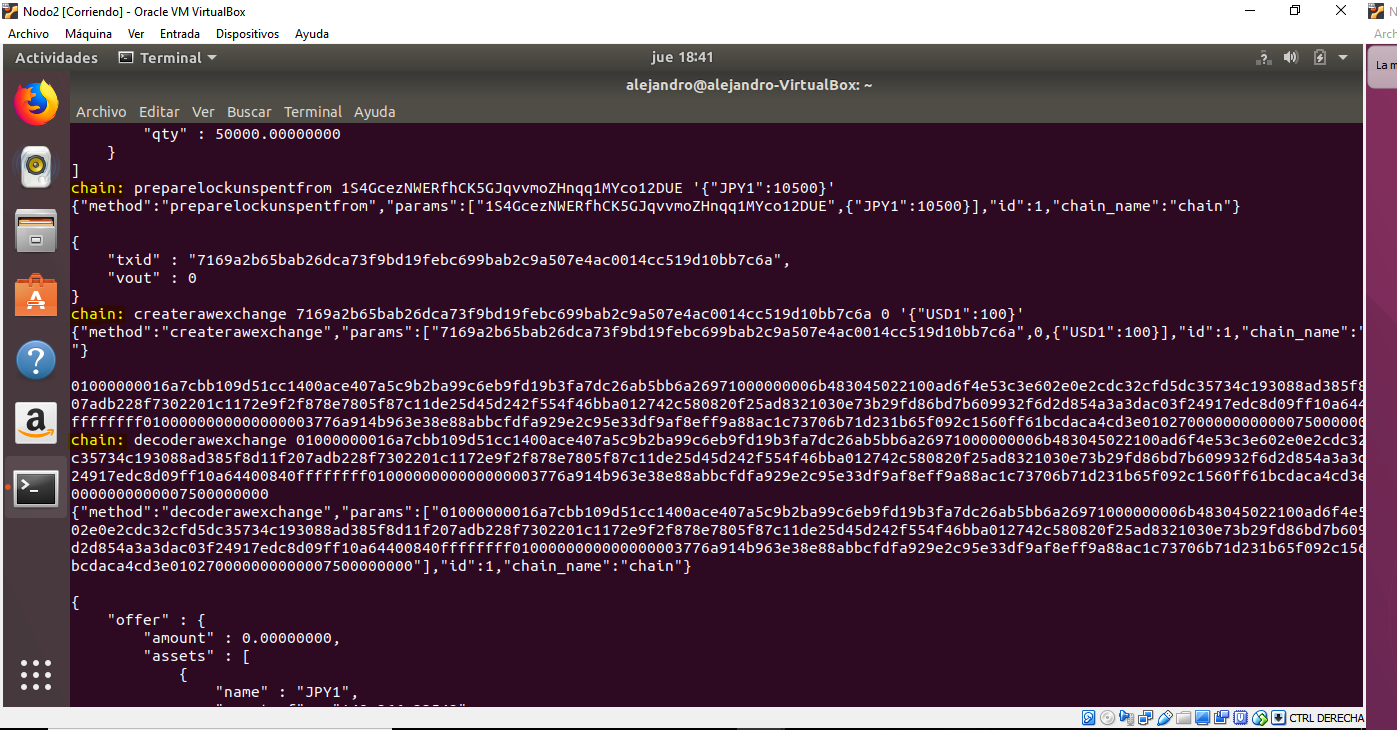
Se ve la dirección en el nodo 2 a la que hay que conceder permisos desde el nodo 1 para poder crear activos. Después de conceder los permisos se crean los activos.

 Listamos los activos que hemos creado:

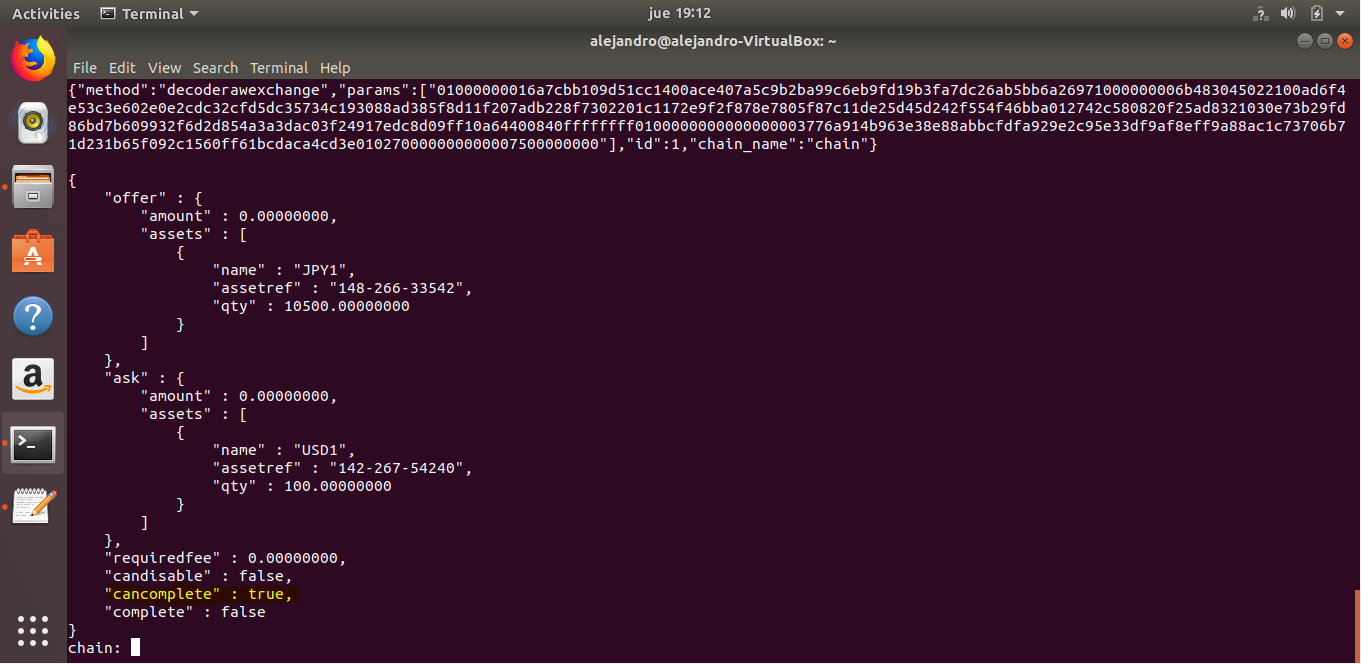
Se realiza get total balance para saber el saldo de ese nodo:



Saldo del nodo 2:

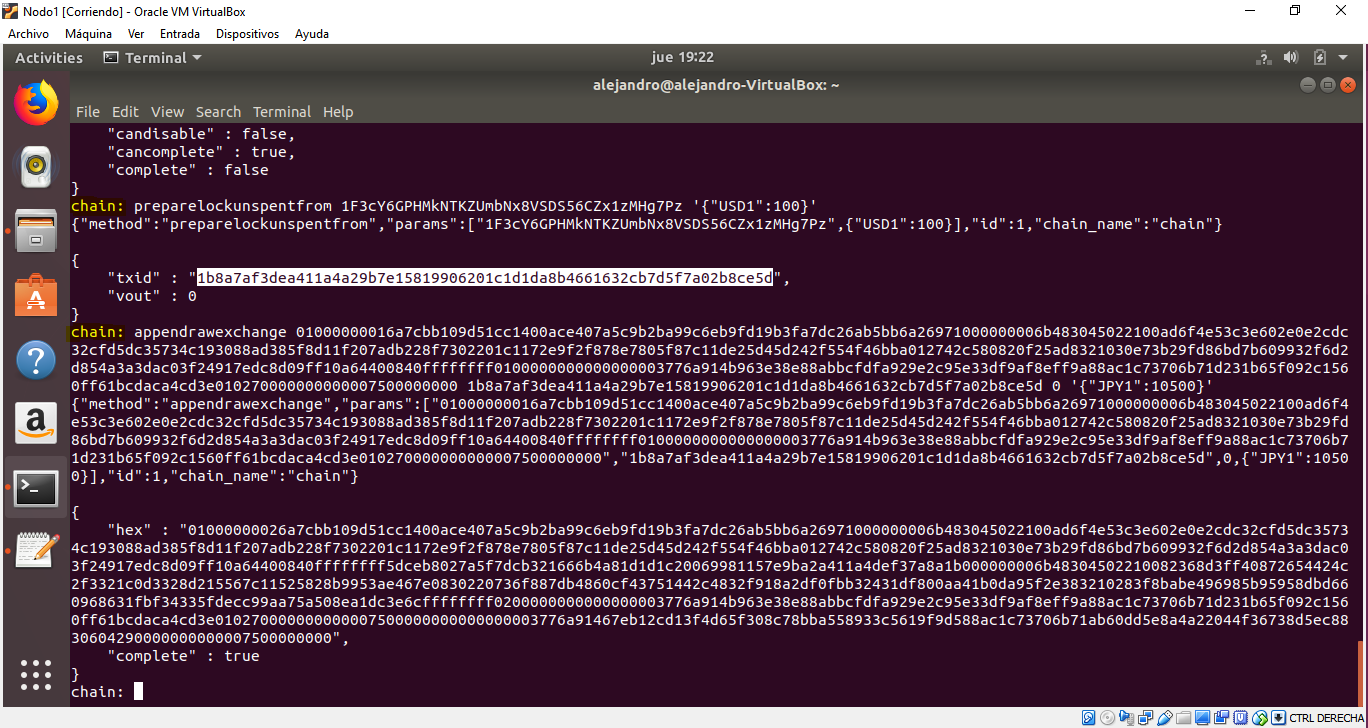
En el nodo 2 creamos una transaccion de 10.500 yenes, pero a cambio pedimos 100 $.

El texto generado en hexadecimal es un id de transacción que contiene los datos de la misma también, y también muestra la información de la transacción

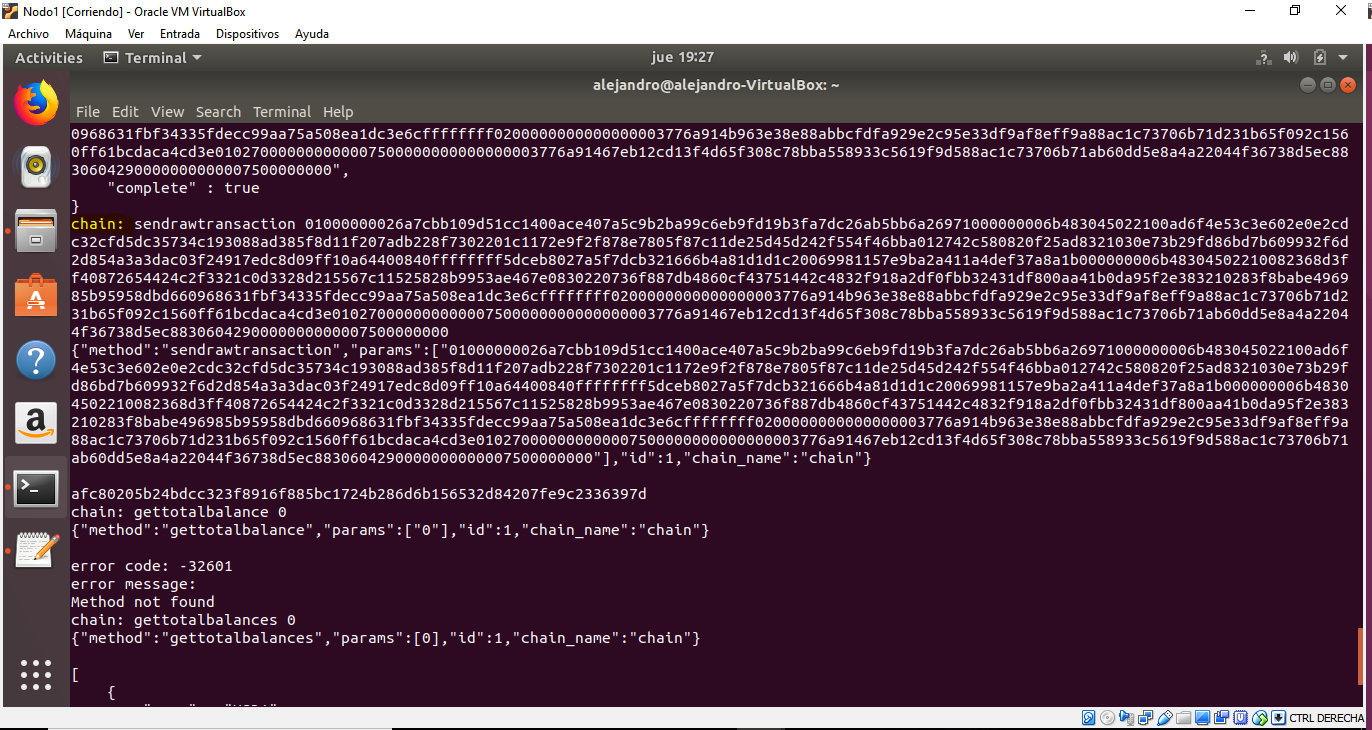
Hay que ejecutar el comando decoderawexchange y ese texto generado para confirmar si el nodo 1 tiene activos suficientes para poder llevarse a cabo la transacción.

Como cancomplete es true significa que tiene suficientes activos.

Ahora crea una salida de transacción bloqueada que contenga 100 dólares, aún en el primer servidor, esto crea otro id. Después hay que agregar esta transacción a la de intercambio, para confirmar que aceptamos esos 10500 yenes



Una vez hecho esto la transacción está firmada y ya solo quedaría que se transmita como hacemos en la siguiente imagen:



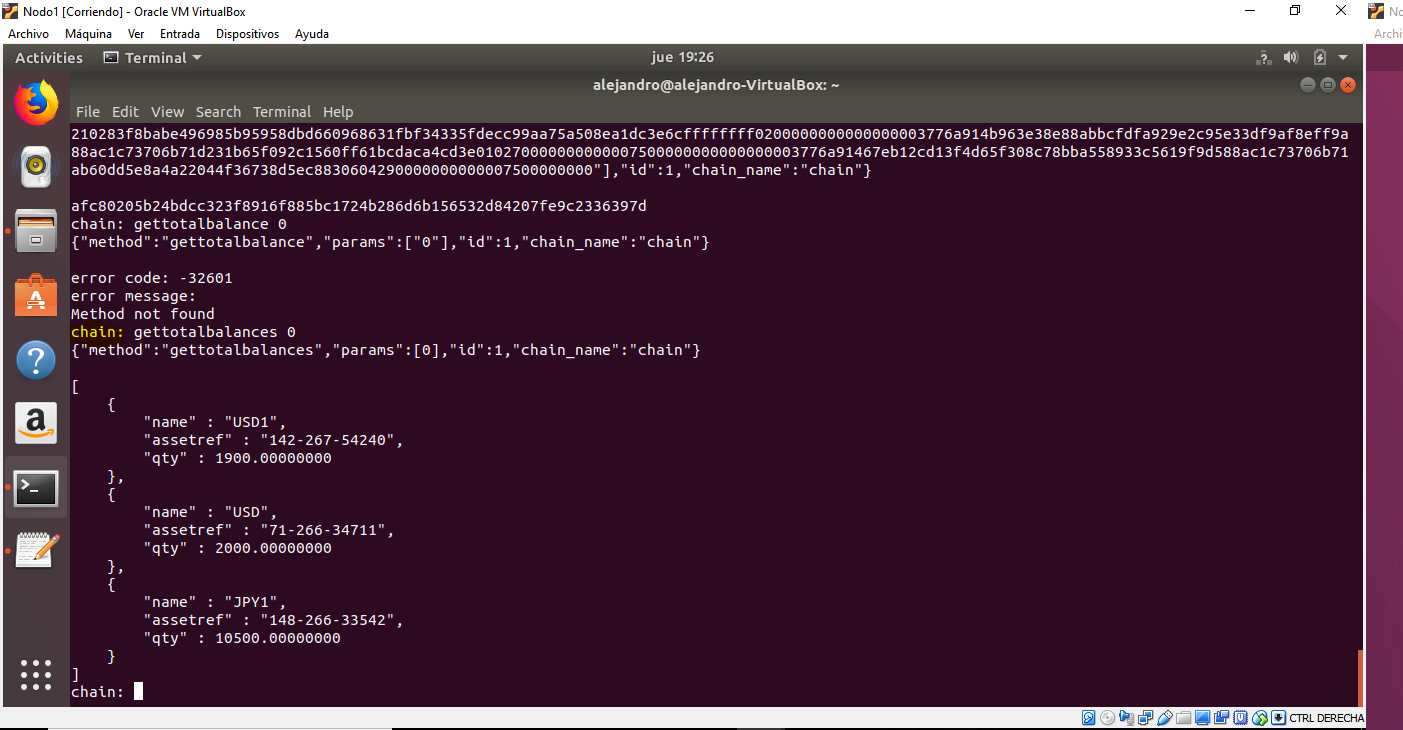
Ya solo queda hacer la comprobación de la transacción que se explicará en otro apartado.

## 4.3 Documentación de pruebas

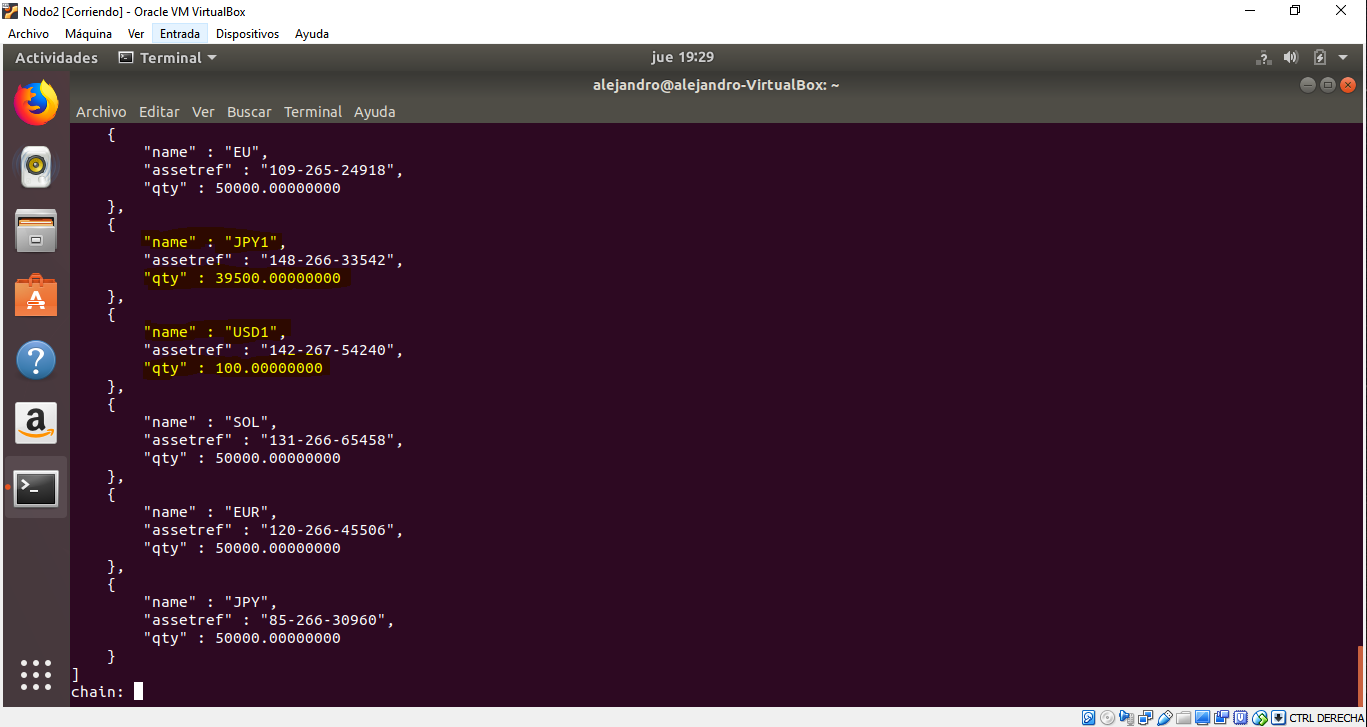
Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los casos de prueba.

### **4.3.1 Comprobaciones**

**Comprobación 1:** Utilizando “gettotalbalances 0” podemos comprobar en el nodo1 todas las monedas creadas para ambos nodos y la cantidad de cada una de ellas en qty:



Se puede comprobar también en el nodo2 de la misma manera. Las dos monedas utilizadas en el ejemplo anterior son JPY1 y USD1 marcadas en amarillo:

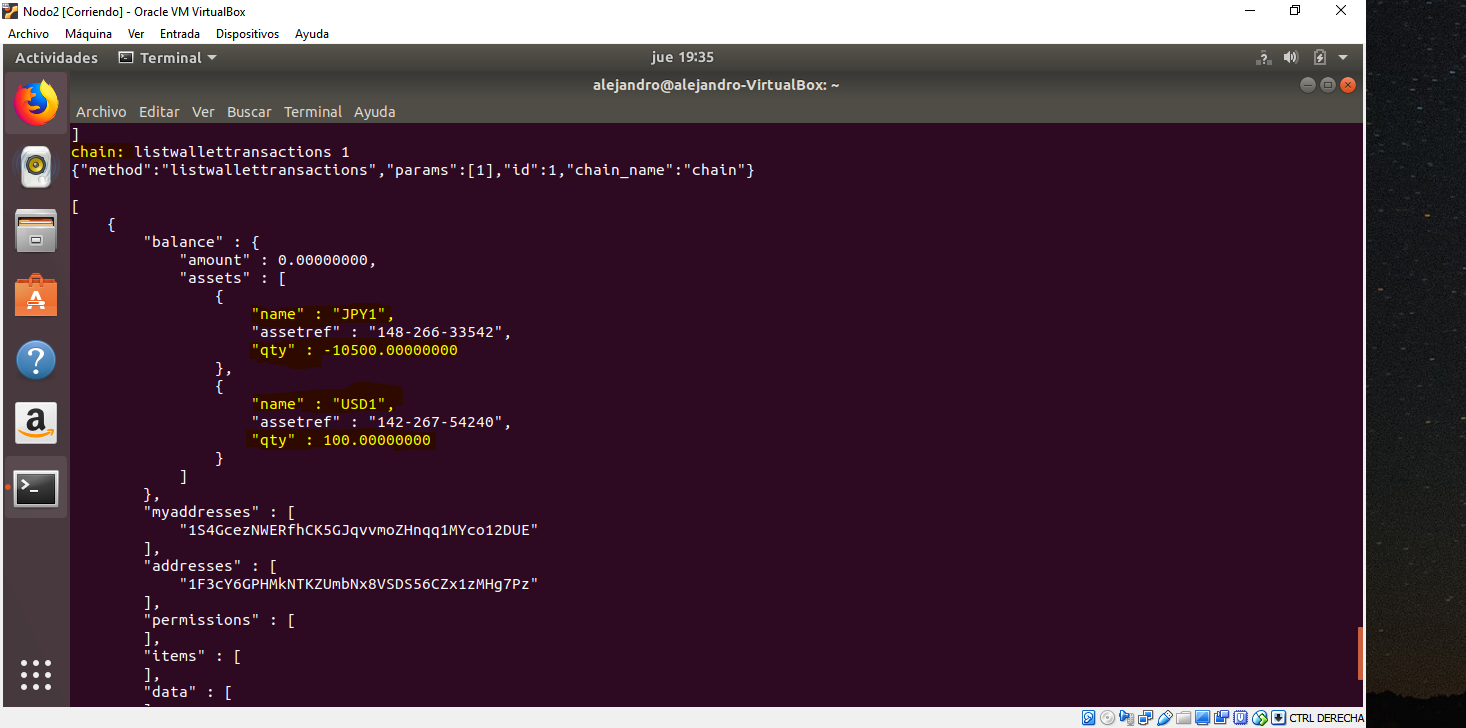


**Comprobación 2:** Utilizando “listwallettransactions 1” podemos ver las monedas con su cantidad pertenecientes explosivamente del nodo donde ejecutamos dicho comando.

Para nodo1:



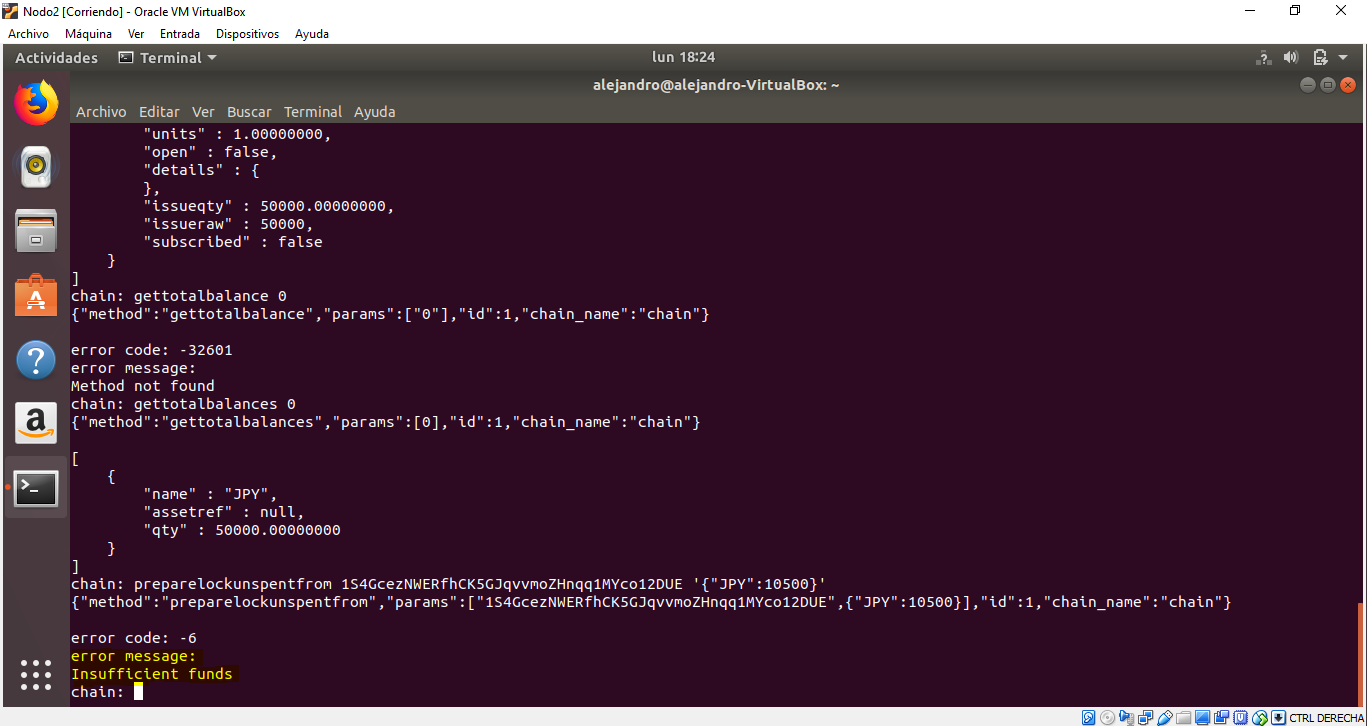
Para nodo2:



### **4.3.2 Errores**

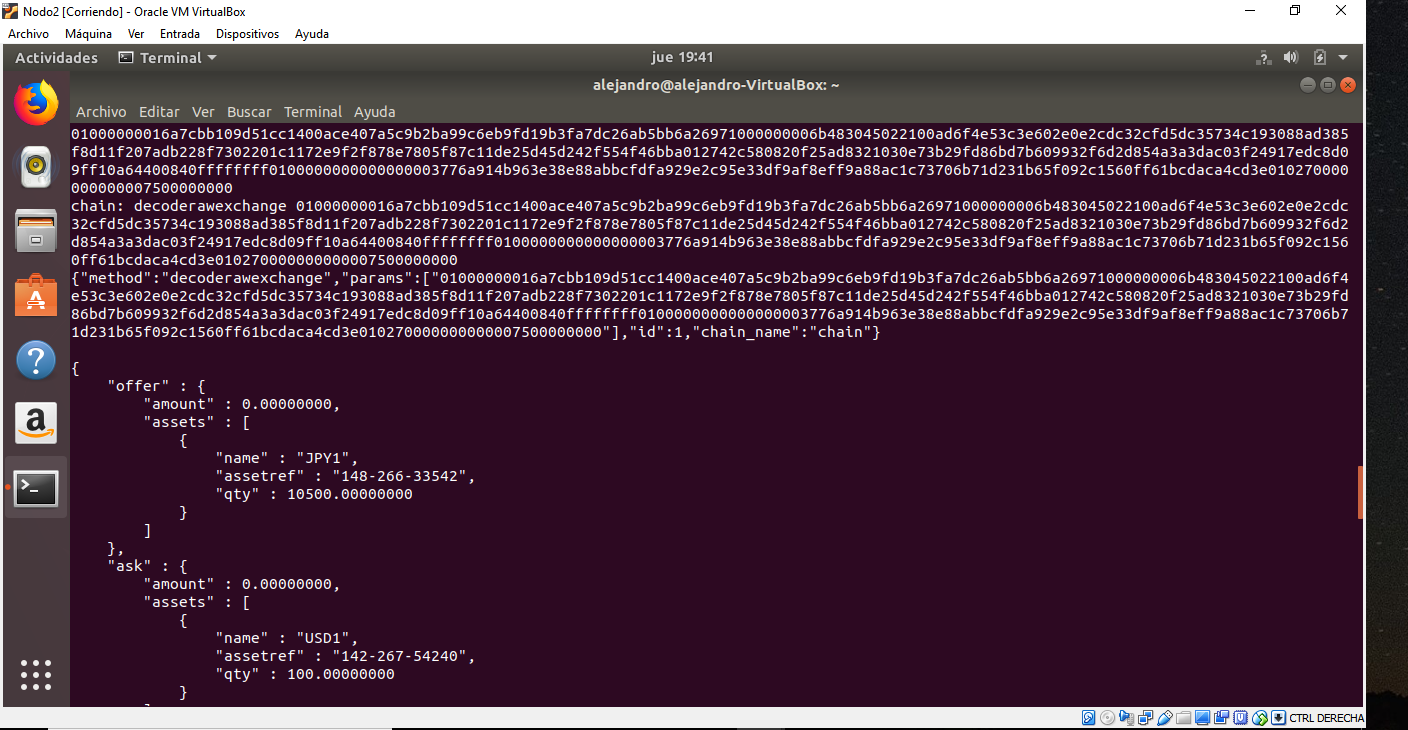
**Error 1:** Al crear la salida de la transacción que contenga 10500 yenes aparece un mensaje de error “insufficient funds” o fondos insuficientes. Esto sucedió a que las JPY creadas no se habían creado bien inicialmente y por tanto no había monedas con las que realizar la transacción.

En caso de introducir una cantidad superior a la cantidad existente el error es el mismo. Se soluciona realizando la transacción con una cantidad inferior (dentro de las existentes) o añadiendo la cantidad suficiente para poder realizar la transacción correctamente.

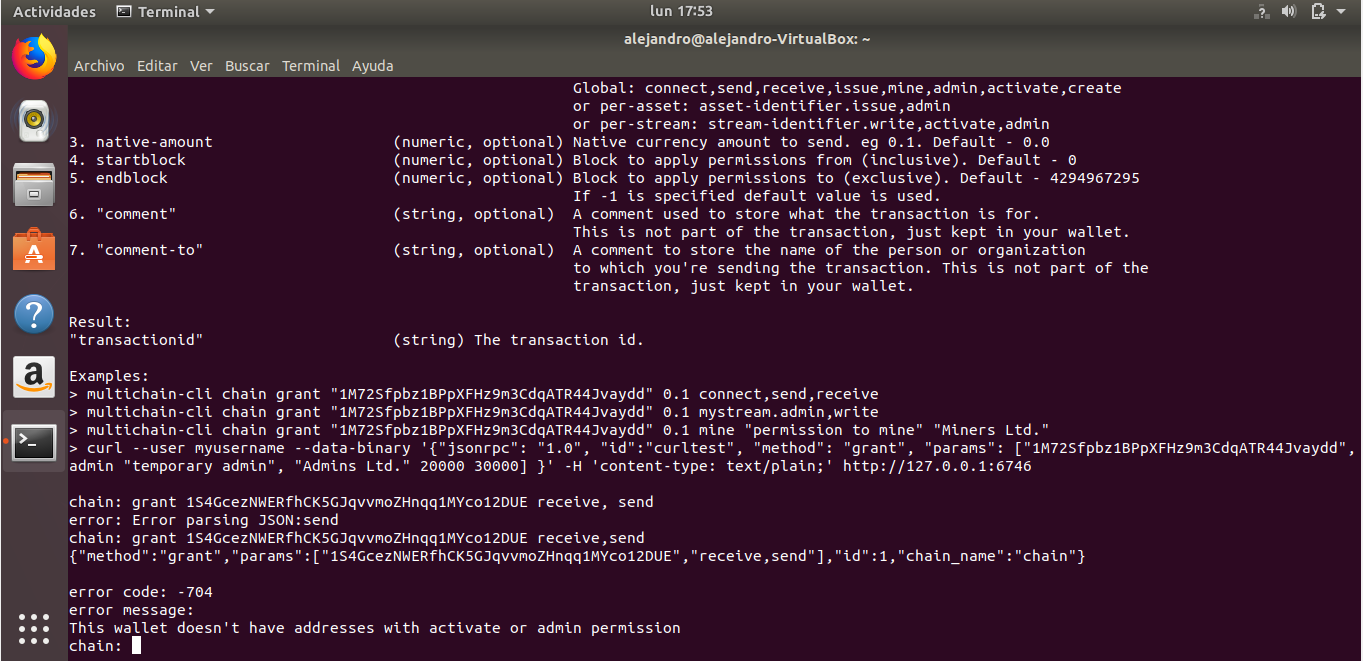


**Error 2:** Al ejecutar “createrwexchange [txid correspondiente] # ‘{“USD”:100}” en el nodo2 se genera un número hexadecimal que contiene los datos de transacción sin procesar que representa la oferta de intercambio. A continuación, hay que ejecutar “decoderawexchange [número hexadecimal]” en el nodo1, pero lo ejecutamos en el nodo2 y se produjo en el nodo1.

Por tanto, se produjo un error que solucionamos copiando el número hexadecimal en un blog de notas que tuvimos que pasar de un terminal a otro y ejecutando el comando decoderaweraxchange en el nodo1 como deberíamos haber hecho inicialmente.



**Error 3:** Realizamos desde el nodo1 la emisión de activos a la dirección del nodo2 de recibir y enviar sin otorgar los permisos previos al nodo1. Se produjo el error 704 y lo solucionamos otorgando los permisos necesarios al nodo1.

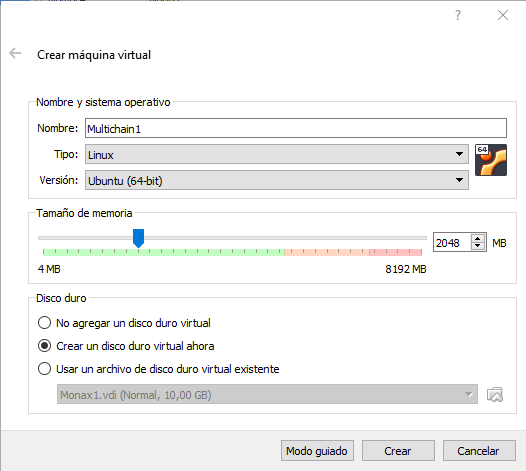


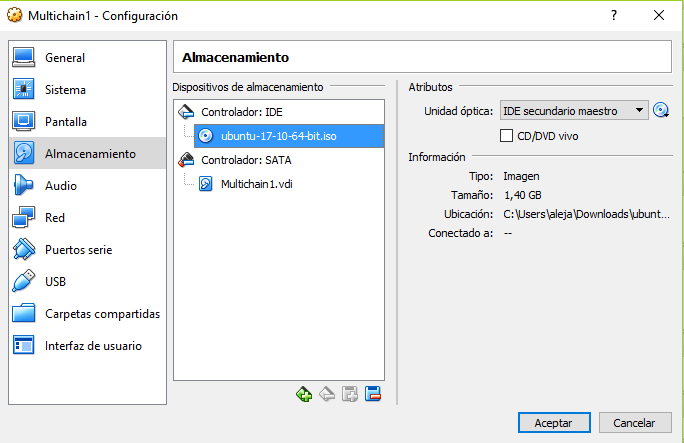
## 4.4 Documentación de instalación

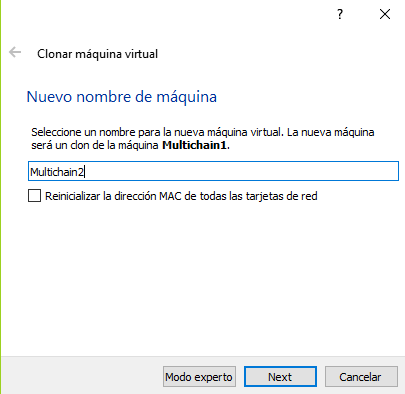
Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

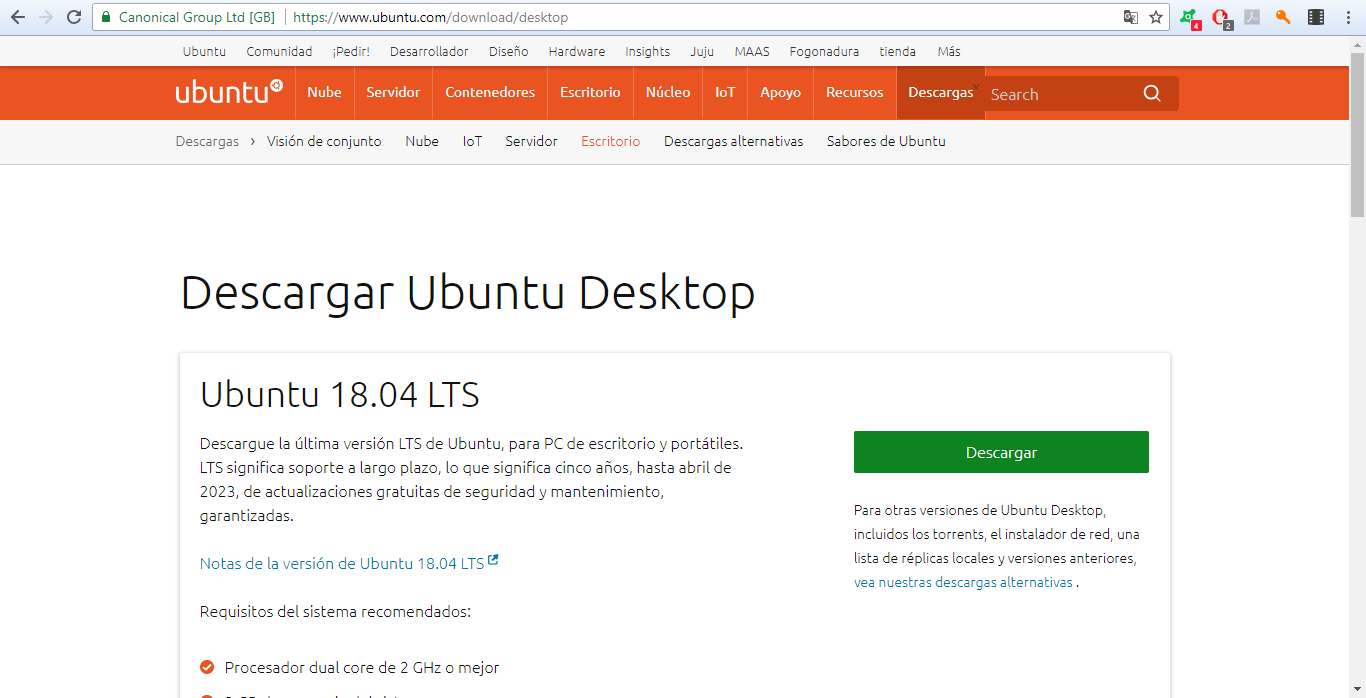
### **4.4.1 Proceso de instalación de las máquinas virtuales**

Se le concede memoria a la máquina virtual, nombre, tipo de sistema operativo etc.







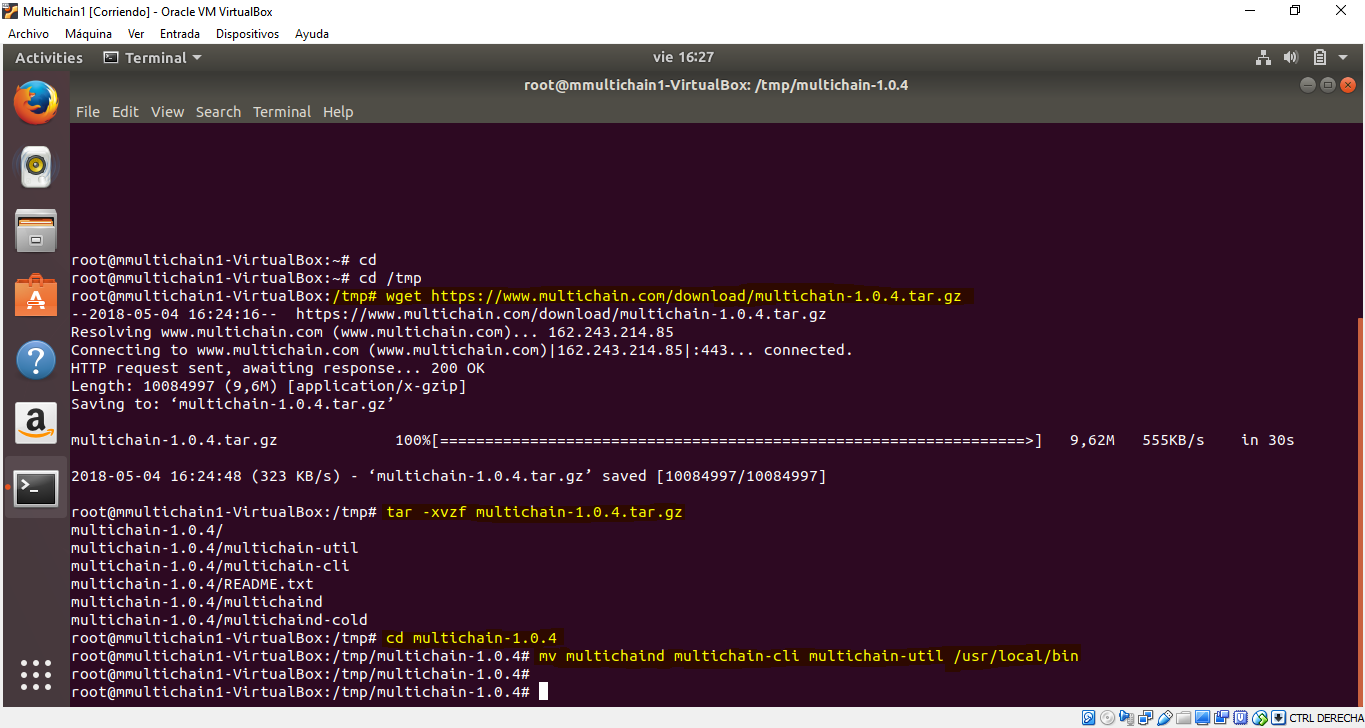


### **4.4.2 Instalación**

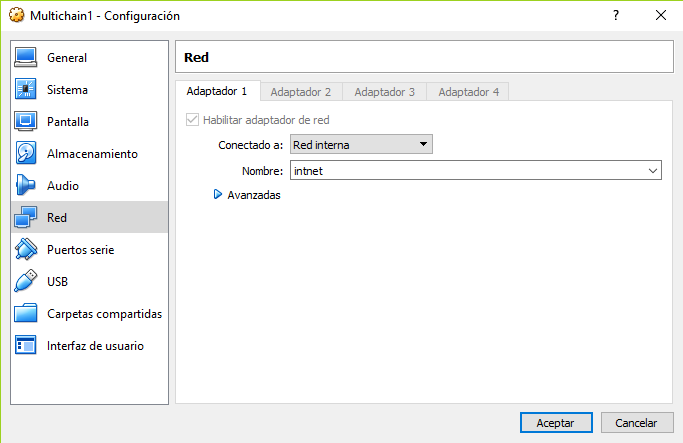
En primer lugar, se descarga el programa en la propia página de MultiChain, dicho programa se descarga mediante comandos posteriormente escritos en el terminal de Linux.

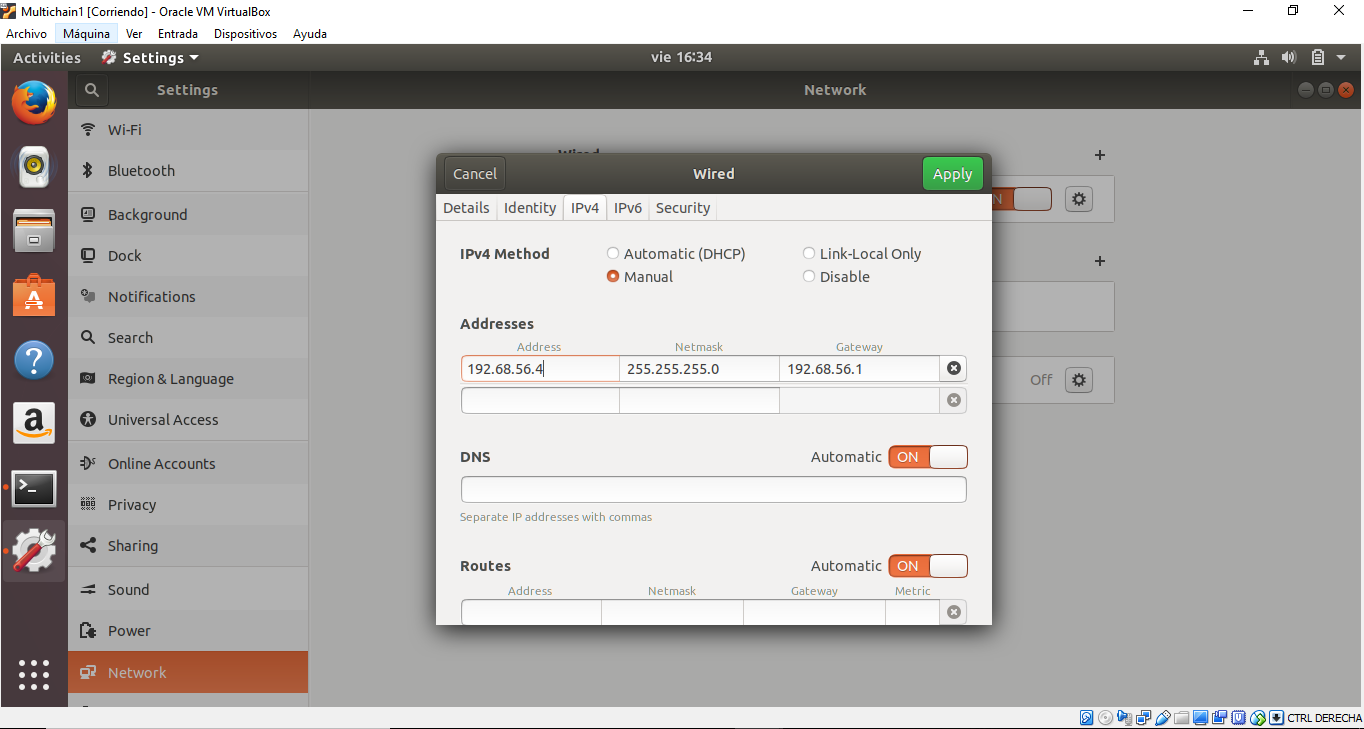
Primero nos descargamos MultiChain con el primer comando subrayado en la siguiente imagen.

El segundo comando seleccionado sirve para descomprimir el archivo descargado, posteriormente nos movemos mediante cd a la carpeta descomprimida y, por último, movemos los archivos multichaind, multichain-cli y multichain-util a usr/local/bin para poder utilizarlos como comandos dentro del terminal Linux, cada uno tendrá una función determinada que será especificada posteriormente.

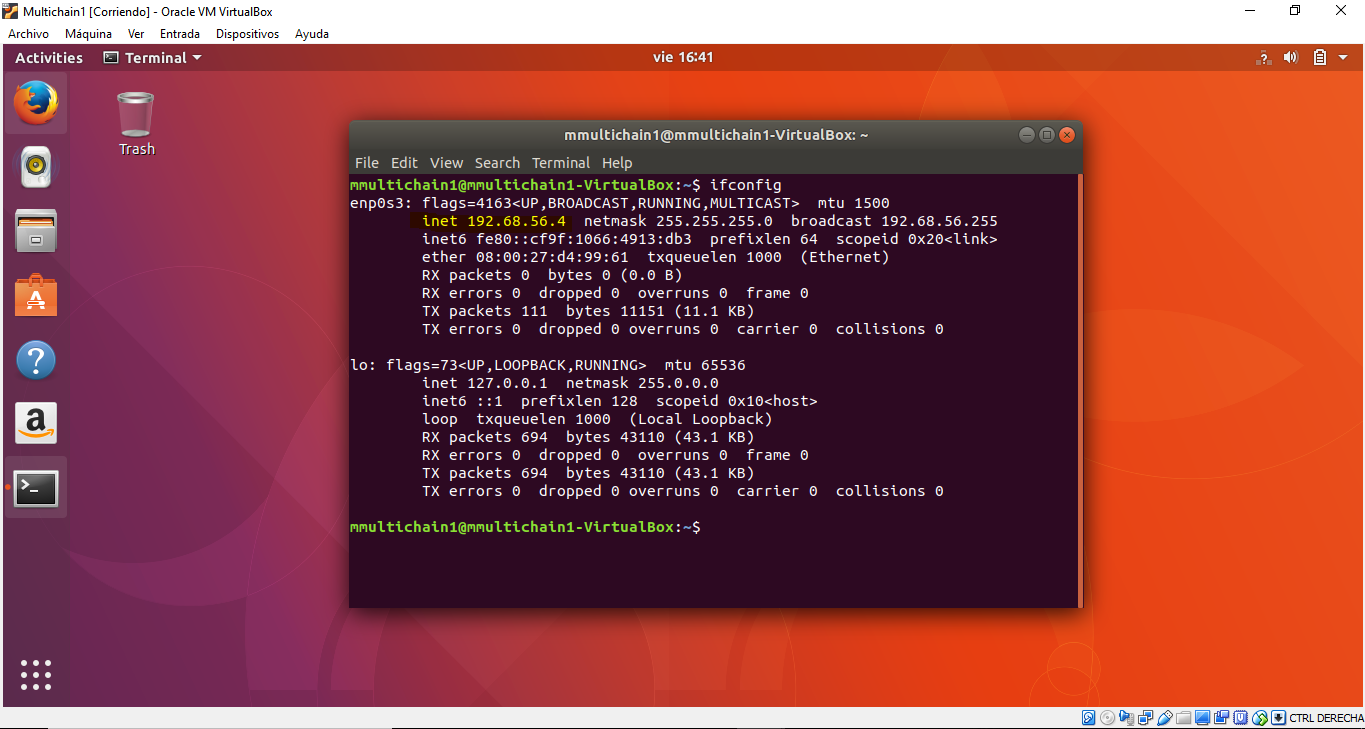


Tras realizar esto, procedemos a conectar el nodo a la red interna para que se pueda ver con el nodo 2, para ello se dota al nodo de una ip como se muestra en la imagen:

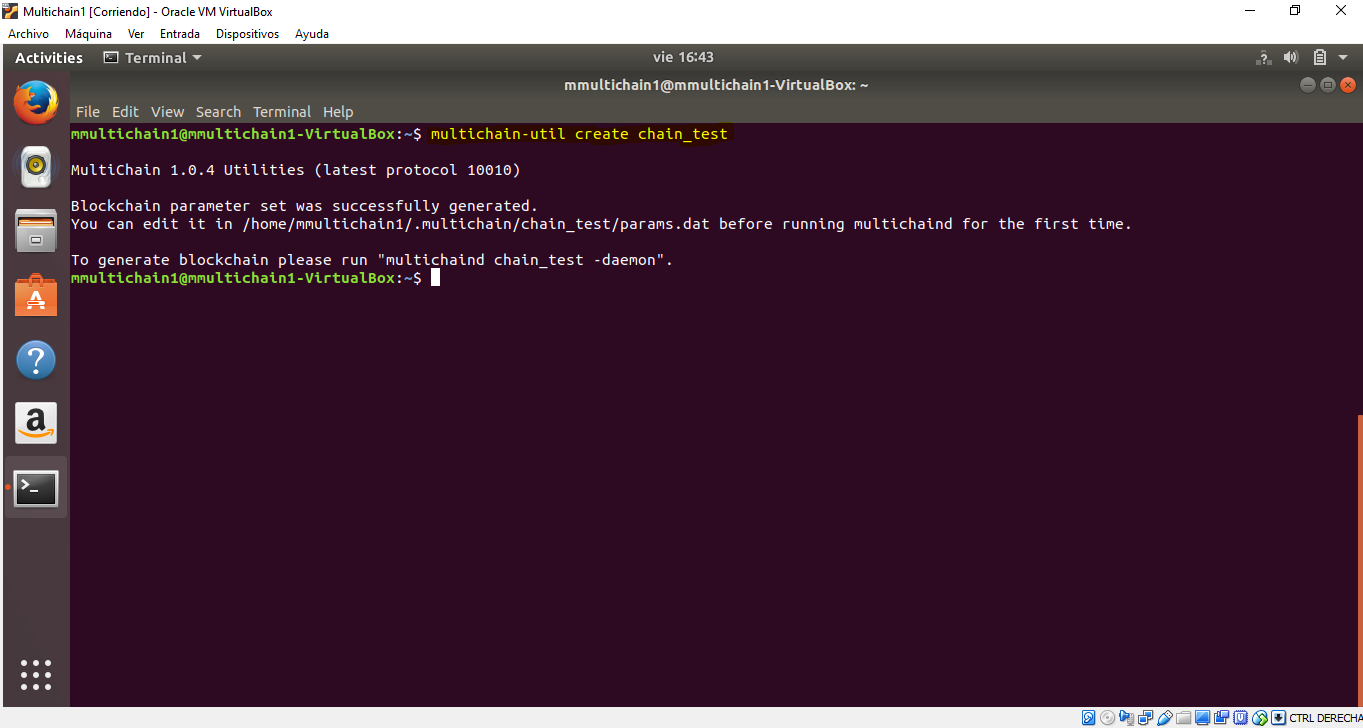




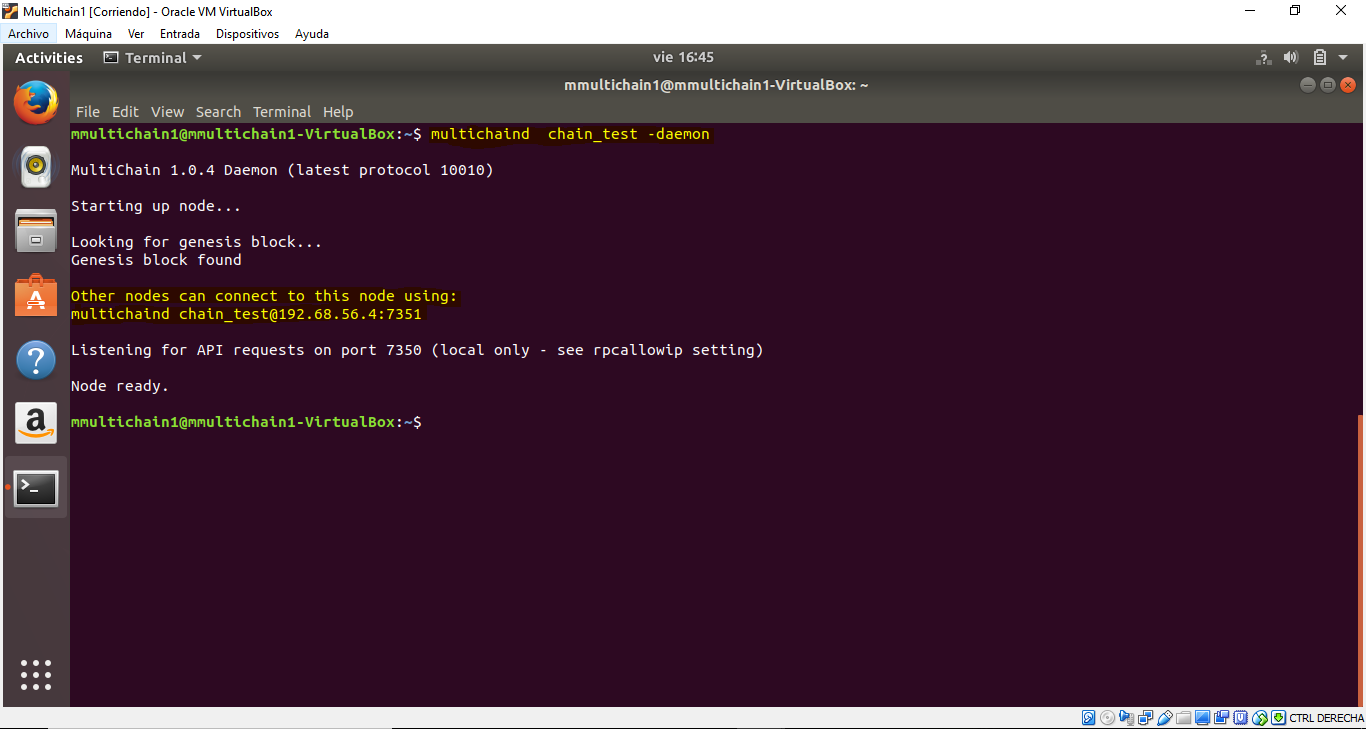
Y comprobamos que la ip se ha cambiado correctamente:



Utilizamos el comando multichain-util create chain\_test para crear la red blockchain:

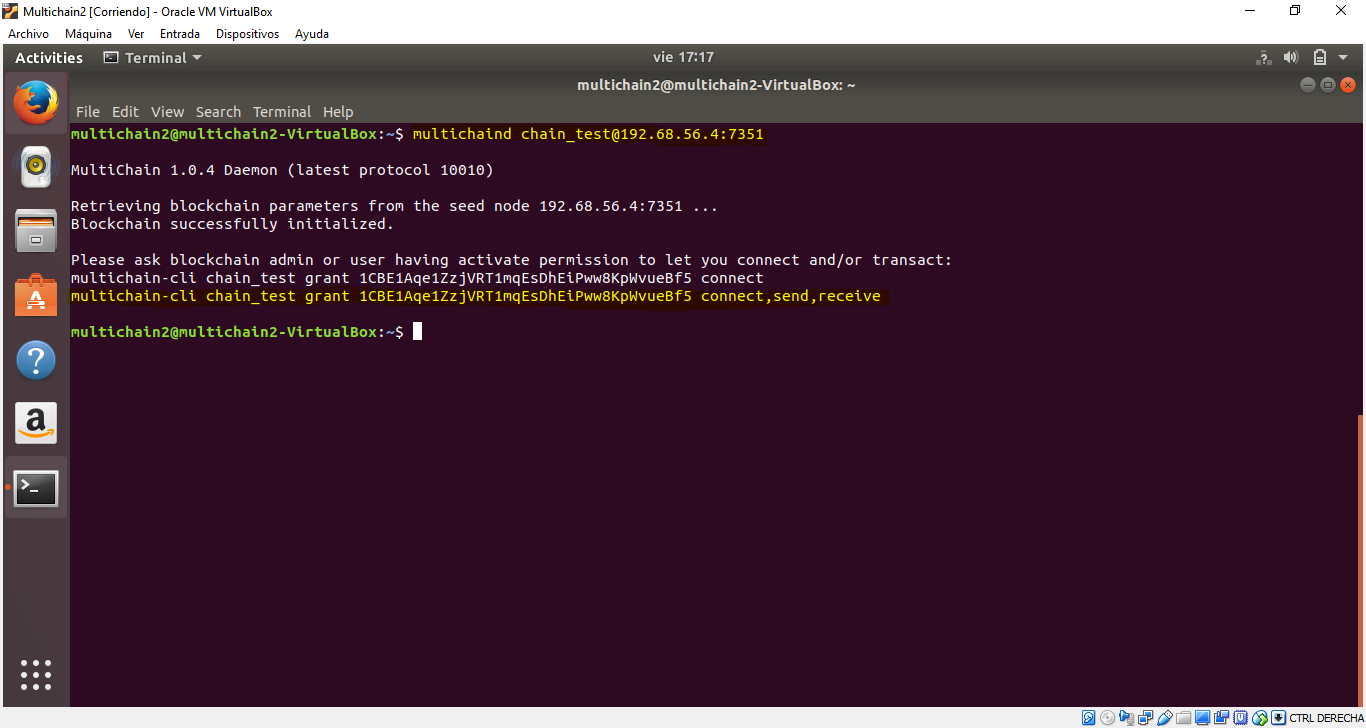


A continuación, con el comando multichaind chain\_test -daemon iniciamos la BlockChain, nos proporciona un método para que otro nodo se pueda conectar a dicha BlockChain, el método se compone del comando multichaind la ip del nodo 1 y el puerto en el que hay que solicitar la conexión:

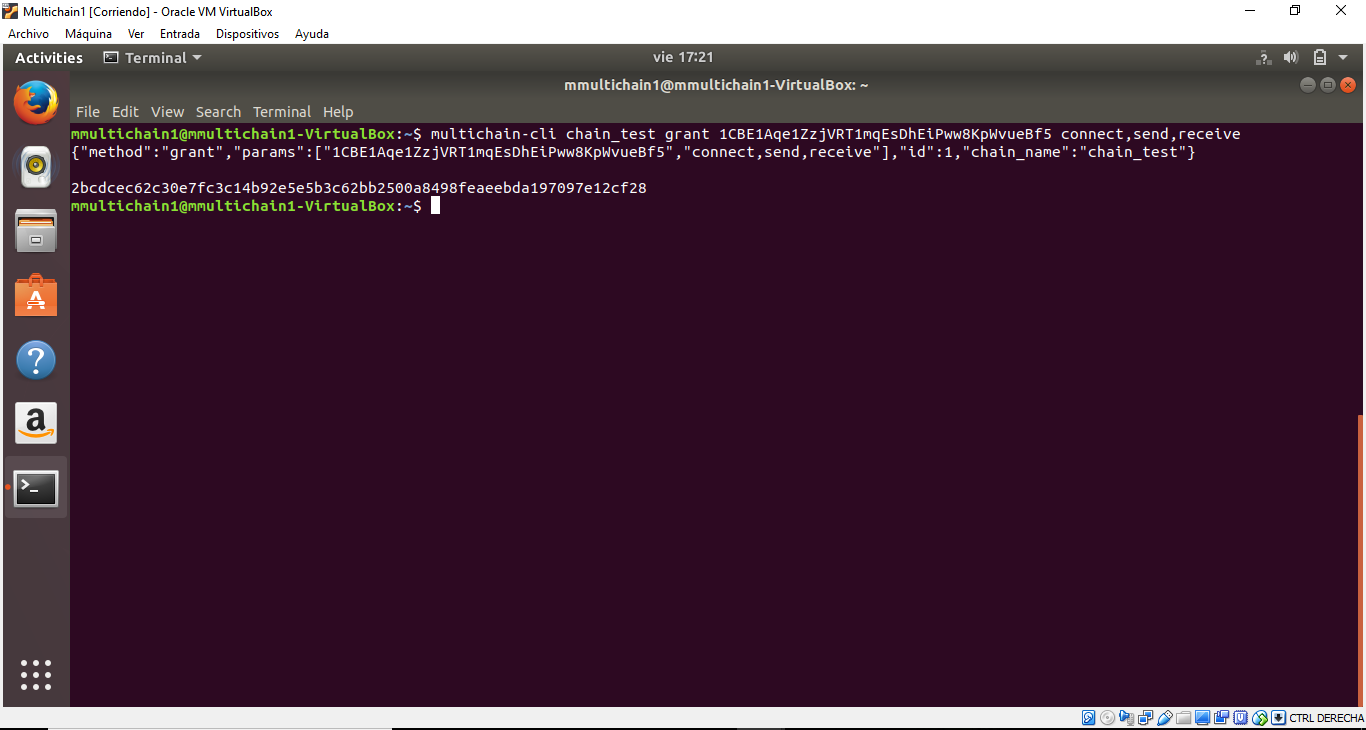


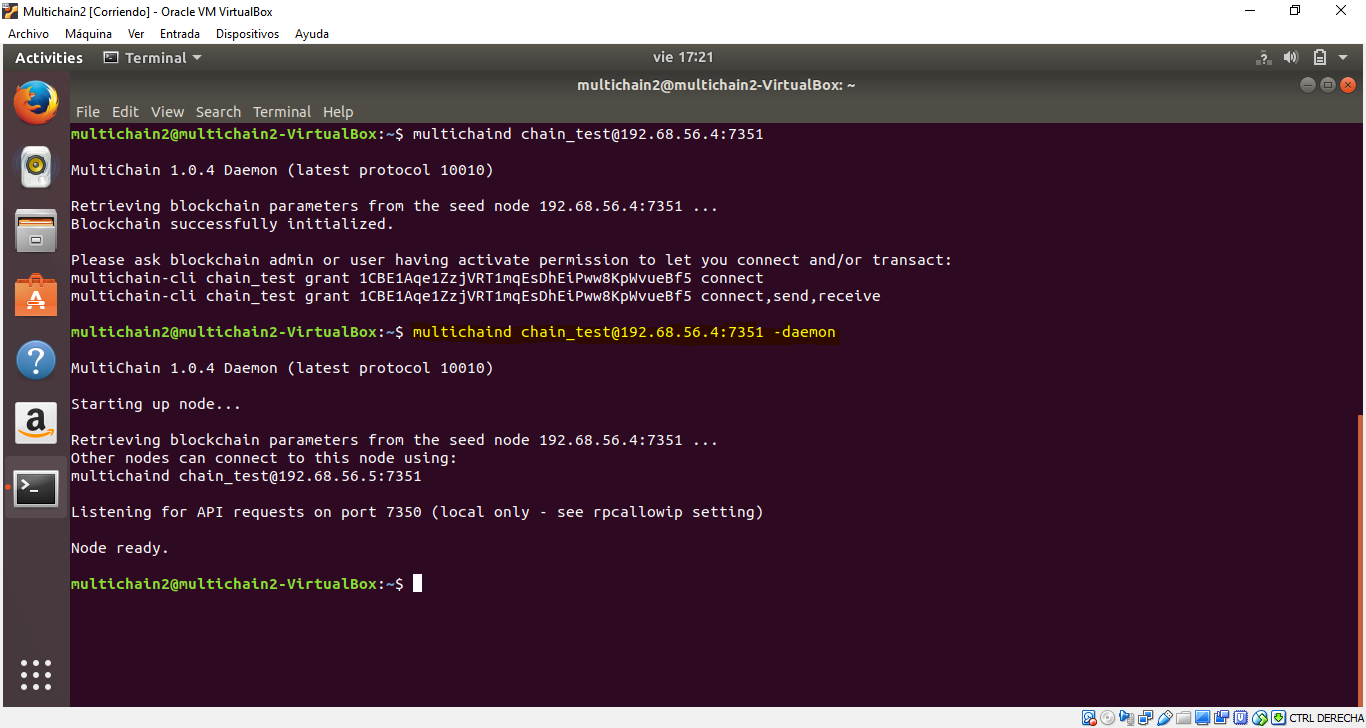
Se repiten todos los pasos anteriores en el nodo 2 a excepción de la creación y conexión de la red BlockChain.

En el nodo 2 se pone siguiente comando para conectar dicho nodo a la red creada anteriormente:



En el nodo 1 se conceden permisos para poder conectar mediante el comando de la siguiente imagen:



Y a continuación, con la información proporcionada por el comando anterior, en el nodo 2 se procede a copiarla para obtener los permisos e iniciar dicho nodo: 

## 4.5 Manual de usuario

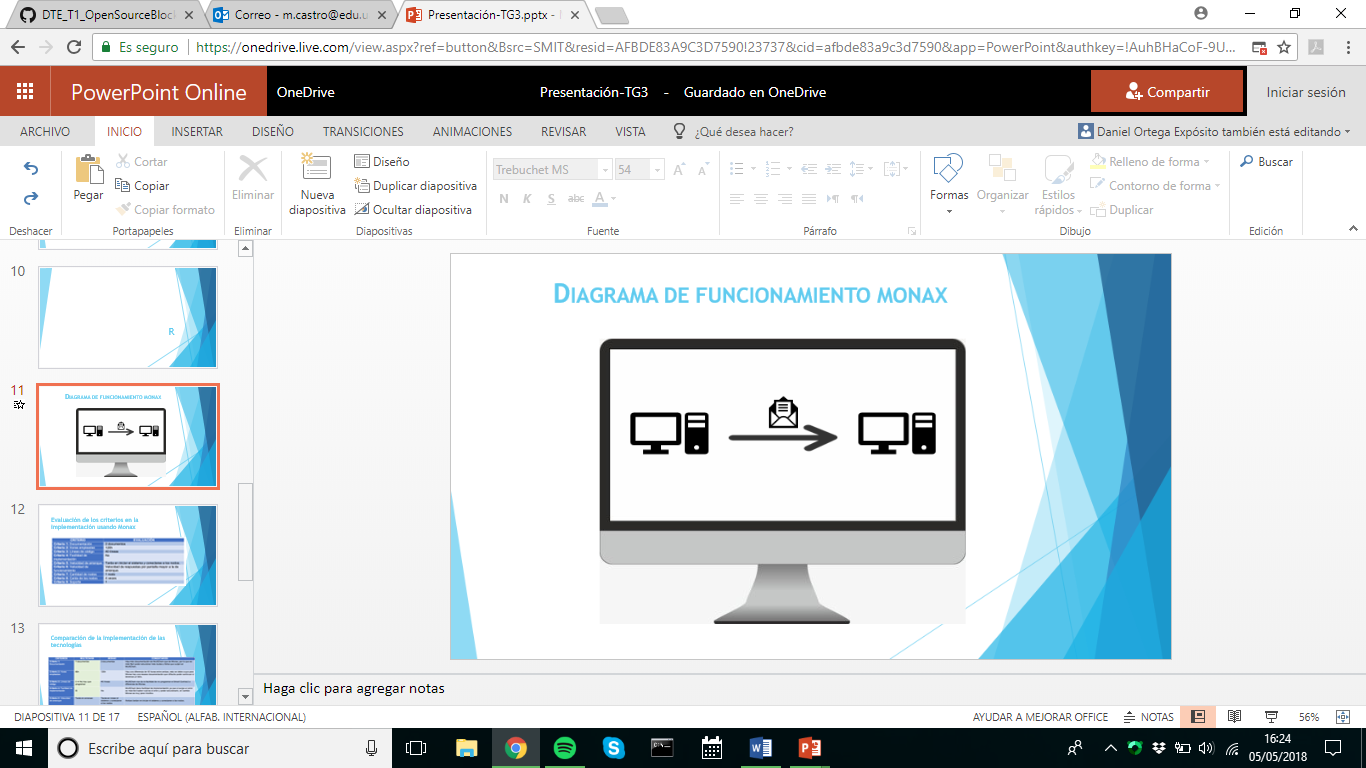
Esta tecnología no está preparada para ser empleada por un usuario que no tenga conocimientos técnicos.

# 5. Proyecto de implementación de un prototipo del sistema utilizando la Monax

Se trata de incluir en este apartado la documentación del desarrollo del proyecto de implementación, utilizando la tecnología B, del sistema cuyos requisitos funcionales se enumeraron en el apartado 2.

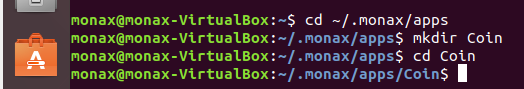
## 5.1 Documentación de diseño

Diagrama de funcionamiento de Monax

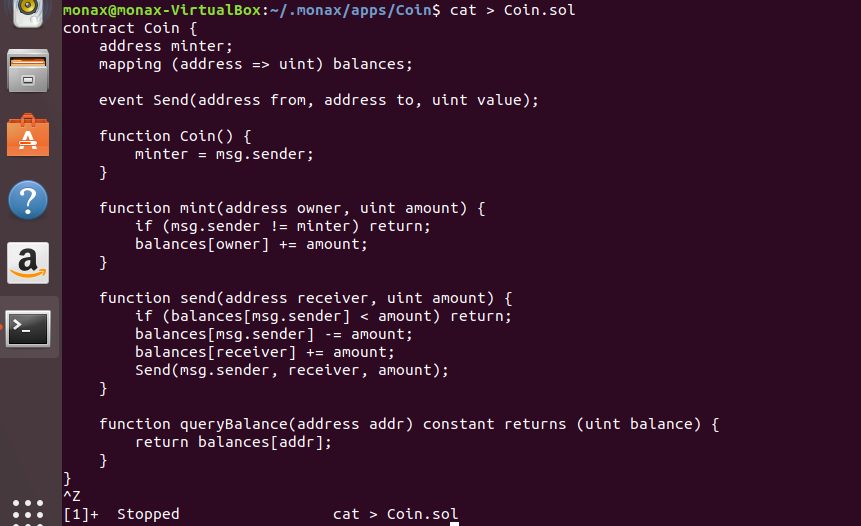


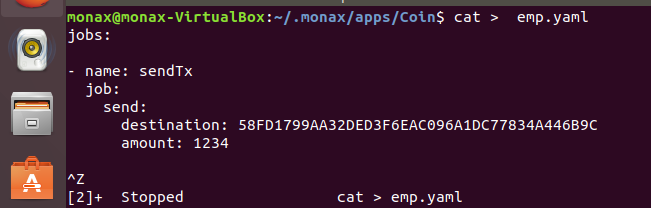
## 5.2 Documentación de construcción

A continuación, explicamos paso a paso la creación de la transacción

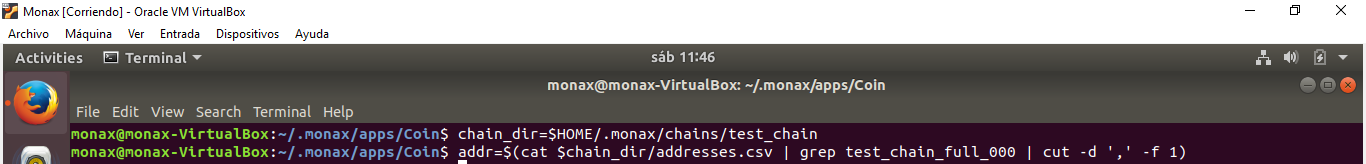


Se crea el archivo correspondiente:





Para poder seleccionar la adress que utilizará la transacción:



Se crea otra red BlockChain llamada receiver\_chain para que interactue con la inicial, siendo creada como la inicial.

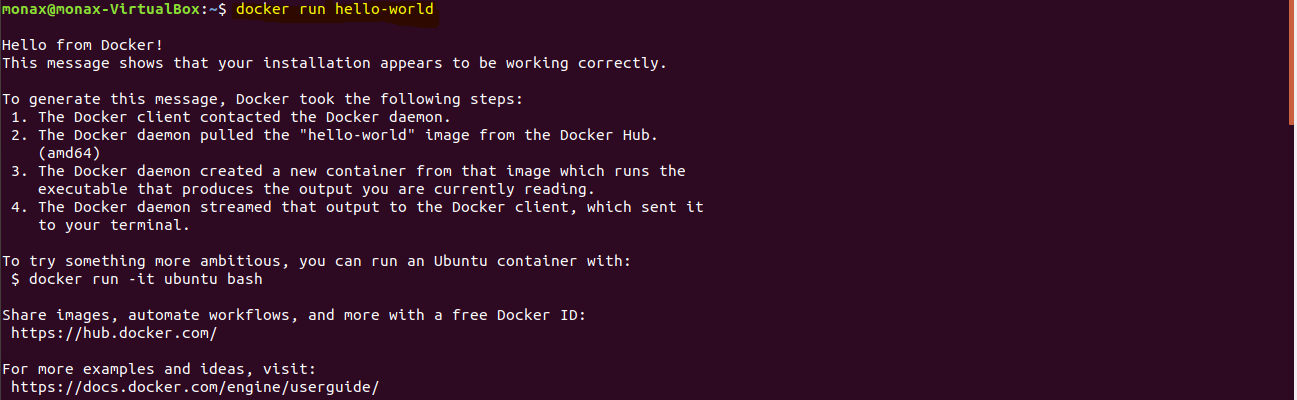
Se ejecuta el contrato entre las dos chains:



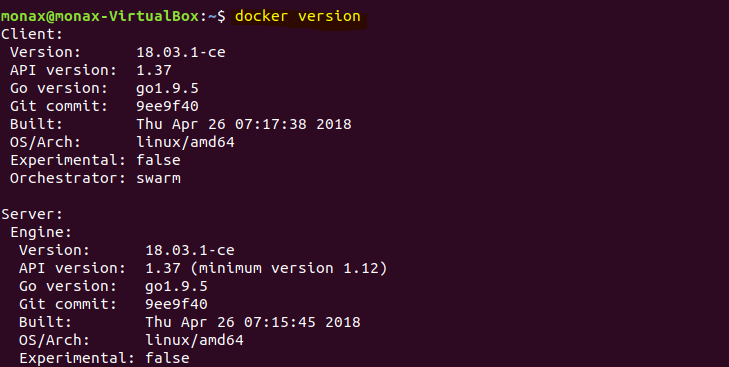
## 5.3 Documentación de pruebas

Casos de prueba establecidos y resultados de las pruebas y acciones de corrección. No es creíble que no hayan aparecido errores en los casos de prueba.

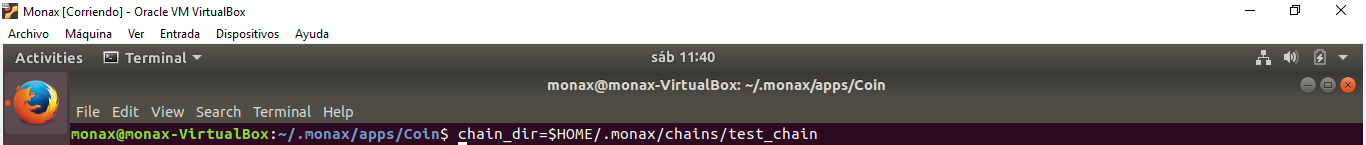
### **5.3.1 Comprobaciones**

**Comprobación 1:** Comprobamos que el nodo se ha conectado bien al Docker

**Comprobación 2:** Comprobamos que el Docker funciona correctamente y muestra el cliente y el servidor

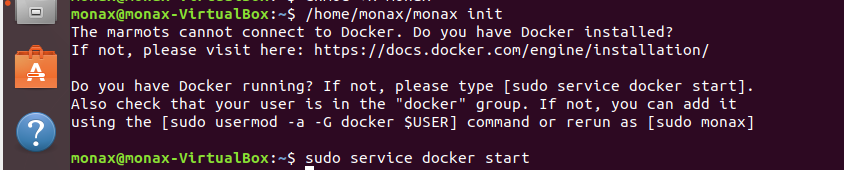


**Comprobación 3:** Comprobamos que encuentra el archivo o directorio y que hemos concedido los permisos correctamente.

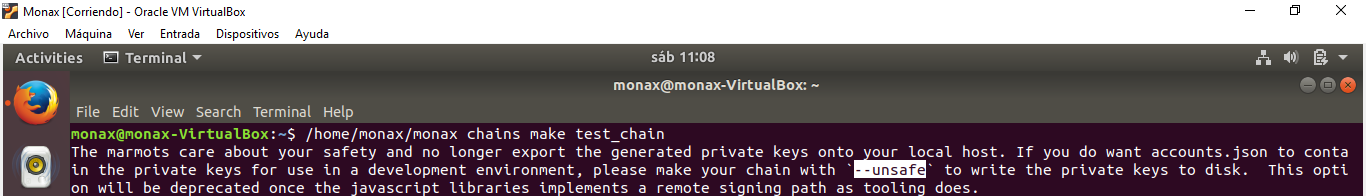


### **5.3.2 Errores**

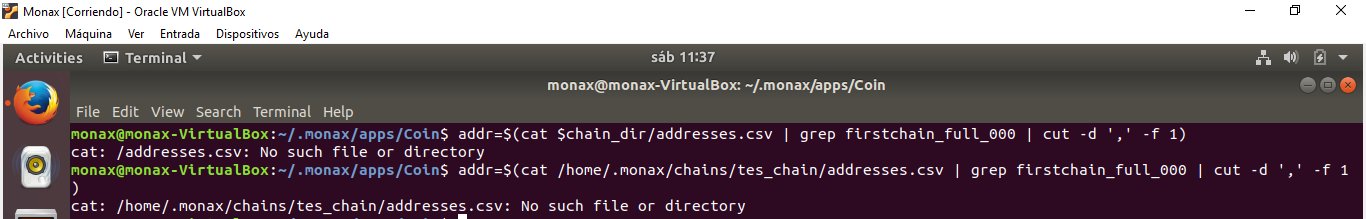
**Error 1:** Al intentar conectarse al Docker se produce un error ya que nos descargamos una versión de Docker no válida.



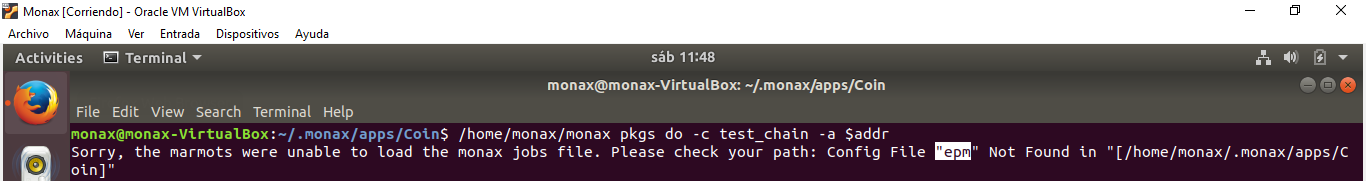
**Error 2:** Se produce un error a la hora de crear la cadena y encontramos otro comando que nos solucionaba este error, que se producía por la que no existía la carpeta en la que se producía la blockchain.



**Error 3:** Se produce un error al no encontrar el archivo directorio y no haber concedido los permisos correspondientes



**Error 4:** Se produce un error al escribir el nombre del archivo y para resolverlo comprobamos el nombre y resolvemos el error.

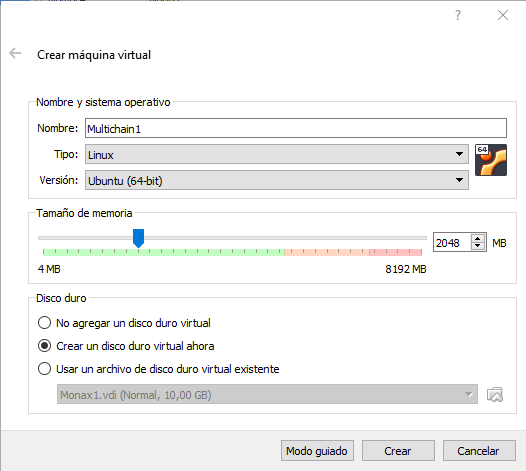


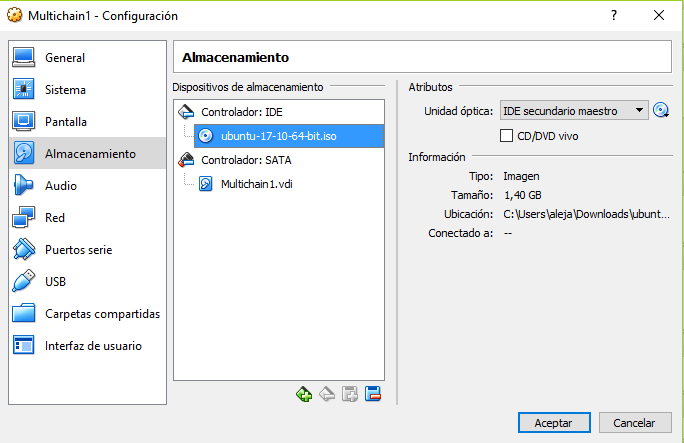
## 5.4 Documentación de instalación

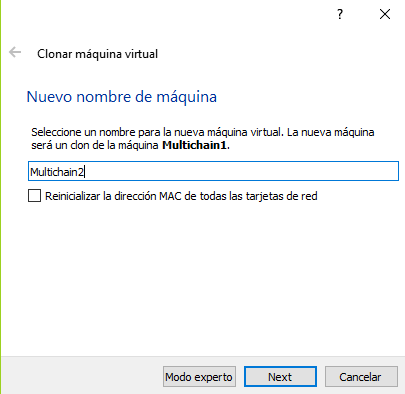
Descripción suficiente para que una persona que no ha participado en el proyecto pueda instalar el prototipo.

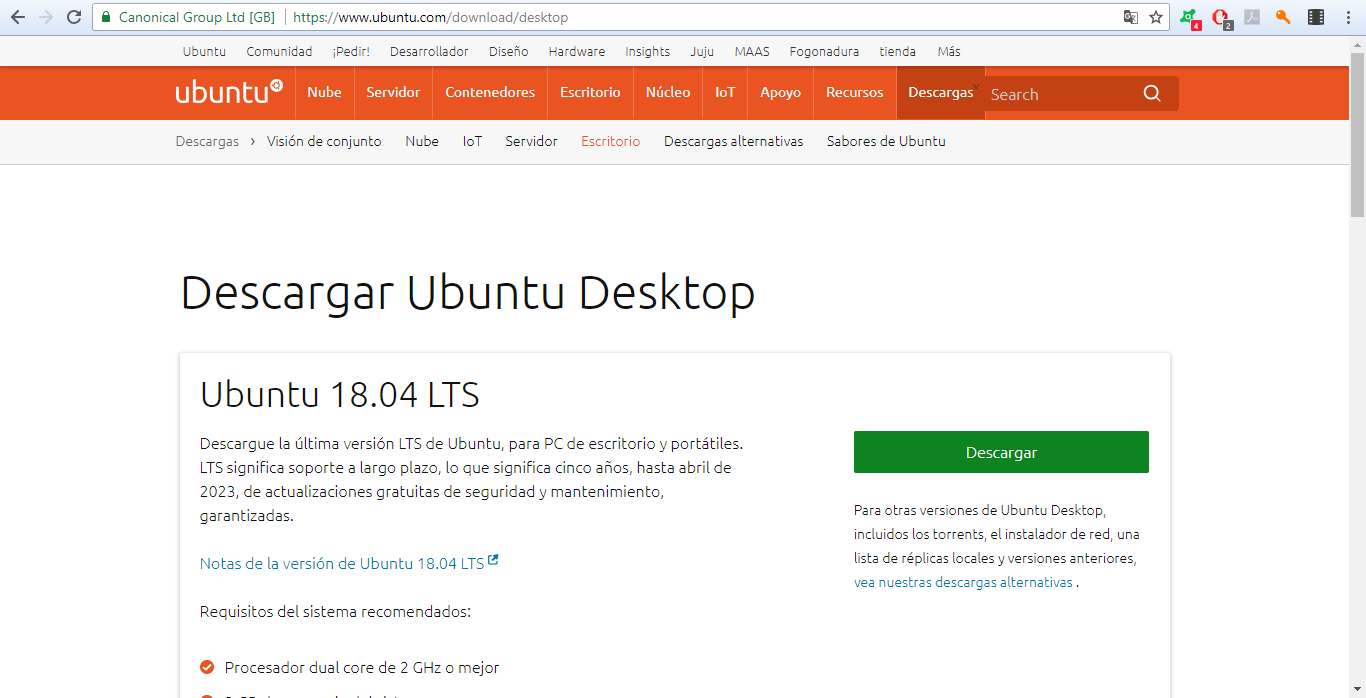
### **5.4.1 Proceso de instalación de las máquinas virtuales**

Se le concede memoria a la máquina virtual, nombre, tipo de sistema operativo etc.







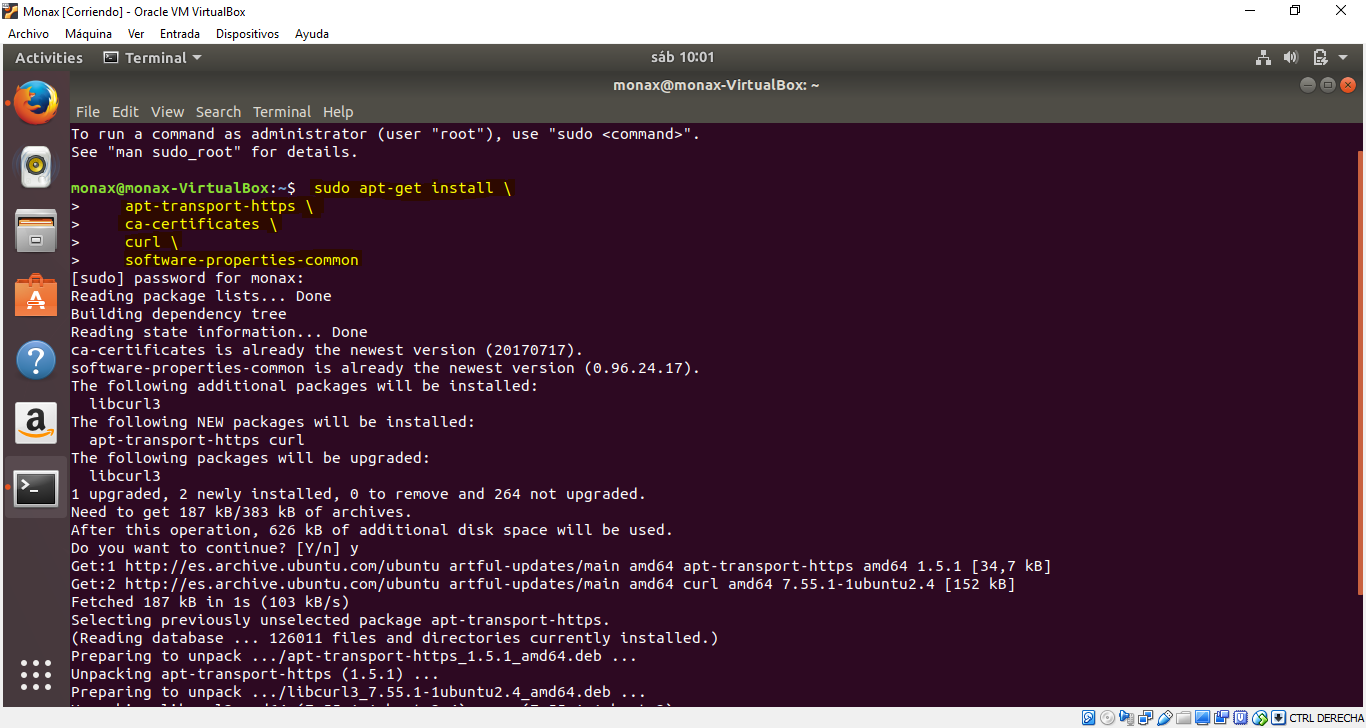
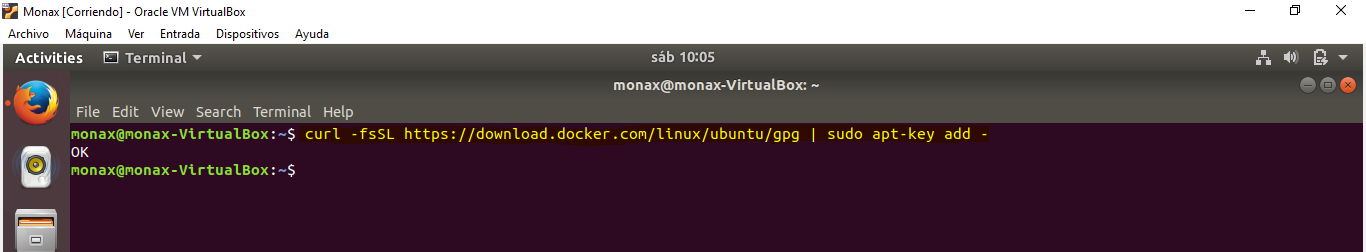


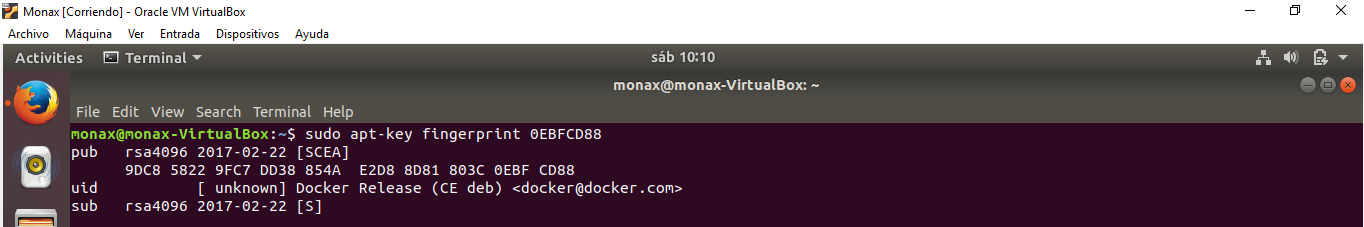
### **5.4.2 Instalación de Docker y Monax**

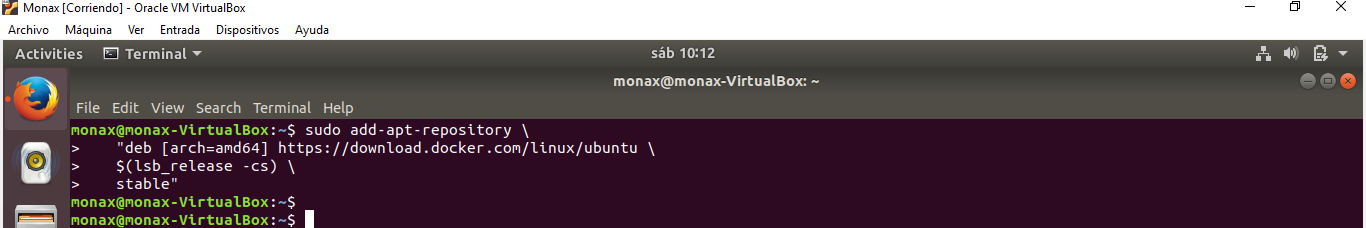
En primer lugar, es necesario instalar el Docker:

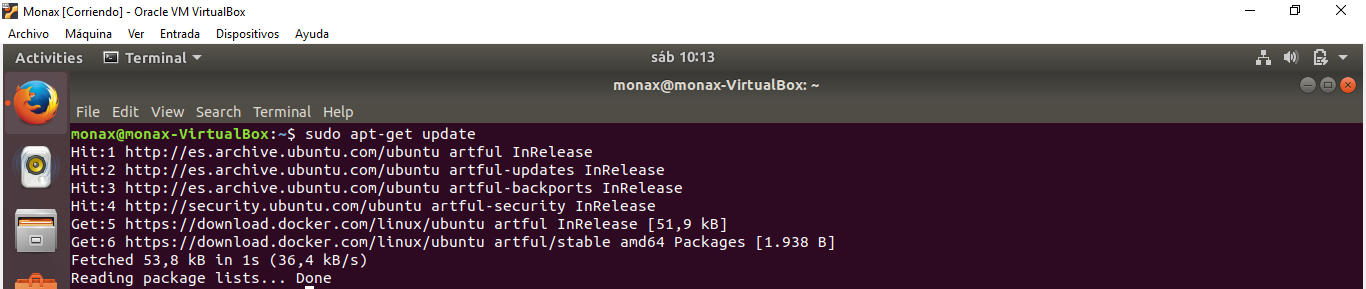
Para descargar Docker entramos en la página: <https://www.docker.com> y descargamos el archivo correspondiente.

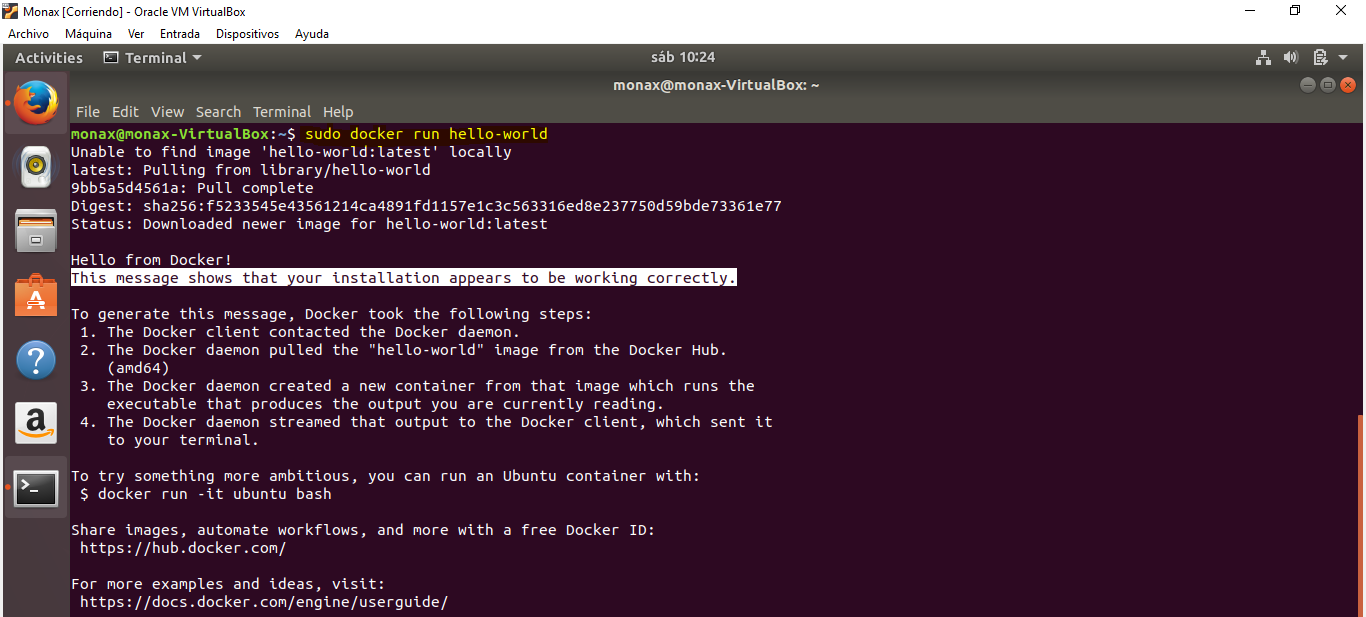
Mediante los siguientes comandos en la consola Linux se procede a la preparación del equipo para instalar el Docker:

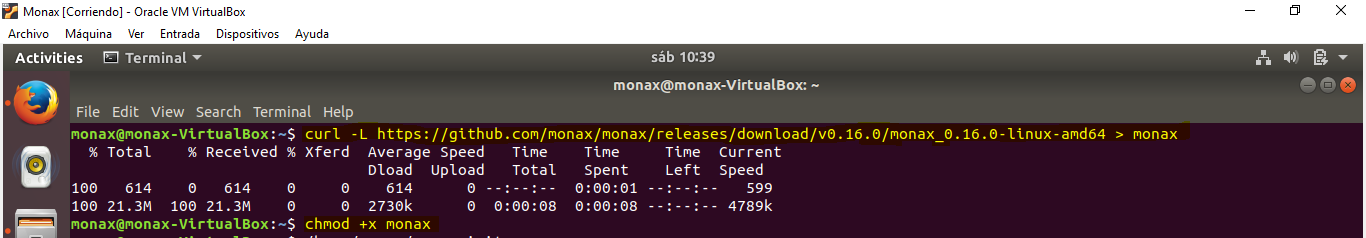




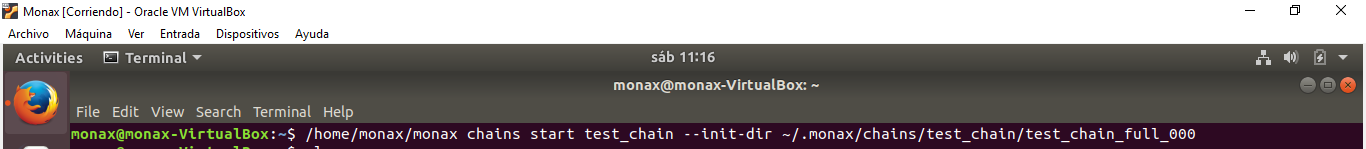


A continuación, las siguientes imágenes muestran la instalación:

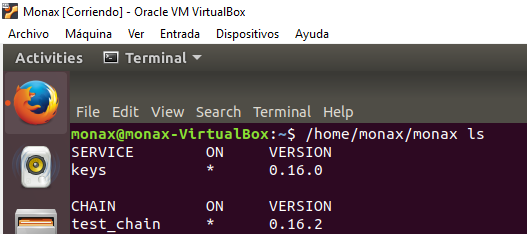
Con el siguiente comando se especifica la versión a descargar:

Tras instalar el Docker, se continua con la instalación de Monax:

Se crea la cadena de prueba:



A continuación, se lista con el comando monax ls para comprobar que está activa:



## 5.5 Manual de usuario

Esta tecnología no está preparada para ser empleada por un usuario que no tenga conocimientos técnicos.

# 6. Comparación de las dos implementaciones

Se trata de dar valores a los criterios de comparación definidos en el apartado 3 sobre la implementación de cada uno de los prototipos.

## 6.1 Evaluación de los criterios en la implementación usando MultiChain

Se incluye una tabla donde se puede observar los criterios de evaluación que se han utilizado, su evaluación y comentarios para su mejor entendimiento.

|  |  |
| --- | --- |
| CRITERIO | EVALUACIÓN |
| Criterio 1: Documentación | 7 documentos |
| Criterio 2: Horas empleadas | 80h |
| Criterio 3: Líneas de código | 0 líneas 🡪 No hay que programar |
| Criterio 4: Facilidad de implementación | Si |
| Criterio 5: Velocidad de arranque | Tarda en arrancar. |
| Criterio 6: Velocidad de funcionamiento | A diferencia de la velocidad de arranque, la velocidad de respuestas por pantalla es mucho mayor. |
| Criterio 7: Cantidad de nodos | 2 nodos |
| Criterio 8: Caída de los nodos | 3 veces |
| Criterio 9: Soporte | 4 |

## 6.2 Evaluación de los criterios en la implementación usando Monax

Se incluye una tabla donde se puede observar los criterios de evaluación que se han utilizado, su evaluación y comentarios para su mejor entendimiento.

|  |  |
| --- | --- |
| CRITERIO | EVALUACIÓN |
| Criterio 1: Documentación | 2 documentos |
| Criterio 2: Horas empleadas | 120h |
| Criterio 3: Líneas de código | 40 líneas |
| Criterio 4: Facilidad de implementación | No |
| Criterio 5: Velocidad de arranque | Tarda en iniciar el sistema y conectarse a los nodos. |
| Criterio 6: Velocidad de funcionamiento | Velocidad de respuestas por pantalla mayor a la de arranque. |
| Criterio 7: Cantidad de nodos | 1 nodo |
| Criterio 8: Caída de los nodos | 4 veces |
| Criterio 9: Soporte | 1 |

# 7. Comparación de la implementación de las tecnologías

Se incluye una tabla resumen donde cruzamos los criterios y los valores de cada tecnología estudiada. También se añade a dicha tabla una columna de comentarios sobre la comparación.

|  |  |  |  |
| --- | --- | --- | --- |
| CRITERIOS | MULTICHAIN | MONAX | COMENTARIOS |
| Criterio 1: Documentación | 7 documentos | 3 documentos | Hay más documentación de MultiChain que de Monax, por lo que es más fácil poder solucionar más dudas y fallos que surjan en MultiChain. |
| Criterio 2: Horas empleadas | 80h | 120h | Hay una diferencia de 40 horas entre ambas, esto se debe a que para Monax hay una escasa documentación que dificulta poder continuar si tenemos un fallo. |
| Criterio 3: Líneas de código | 0 líneas 🡪 No hay que programar | 40 líneas | MultiChain nos da la facilidad de no programar el Smart Contract a diferencia de Monax. |
| Criterio 4: Facilidad de implementación | Sí | No | MultiChain tiene facilidad de implementación ya que si surge un error es más fácil saber cuál es el error y poder solucionarlo, en cambio Monax es muy poco intuitivo. |
| Criterio 5: Velocidad de arranque | Tarda en arrancar. | Tarda en iniciar el sistema y conectarse a los nodos. | Ambas tardan en iniciar el sistema y conectarse a los nodos. |
| Criterio 6: Velocidad de funcionamiento | A diferencia de la velocidad de arranque, la velocidad de respuestas por pantalla es mucho mayor. | Velocidad de respuestas por pantalla mayor a la de arranque. | Ambas tienen una velocidad de respuesta muy parecida. |
| Criterio 7: Cantidad de nodos | 2 nodos | 1 nodo | Monax al usar solo 1 nodo tiene un menor tiempo de implementación, en cambio en MultiChain se tienen que implementar 2 nodos. |
| Criterio 8: Caída de los nodos | 3 veces | 4 veces | Los nodos de MultiChain se han caído 3 veces a diferencia de Monax que se han caído 4 veces. |
| Criterio 9: Soporte | 4 | 1 | MultiChain cuenta con un mejor soporte que Monax.  MultiChain tiene un foro de preguntas y en el caso de Monax tienen un chat de preguntas, MultiChain tiene un tiempo de respuesta máximo de 1 día y te intentan solventar fallos y dudas, en cambio Monax, además de tener que hacer las preguntas en inglés, tardan en responder o no responden directamente. |

# 8. Conclusiones

A partir de lo expuesto en apartados anteriores y tras la realización de las aplicaciones con ambas tecnologías concluimos que está tecnología es difícil de implementar si no eres experto.

Tras utilizar las dos plataformas y realizar la comparación, concluimos con que la plataforma MultiChain es más intuitiva, además de estar mejor documentada que Monax. También proporciona mayores facilidades en cuanto a instalación e implementación. Como ventaja destacable de Monax es que no se necesita nada más que un nodo para la realización de la red blockchain.

Cabe destacar que Multichain tiene integrados los Smart Contracts en sus comandos por lo que para realizar una transacción sólo hay que poner un comando y el Smart Contract que tiene asociado ese comando, en una capa más interna, se ejecutará y se producirá la transacción.

Por último, cabe destacar que MultiChain ofrece un soporte mayor al que ofrece Monax.

# ANEXO 1: Otros Requisitos

## A.1. MultiChain

### A.1.1. Requisitos Funcionales

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
| RF01 | Conectado a: Red interna |
| RF02 | Se debe llevar a cabo una transacción entre dos nodos o más |

### A.1.2. Requisitos NO Funcionales

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
| R01 | Se utilizará la versión 1.0.4 de MultiChain |
| R02 | La implementación de la BlockChain y el desarrollo de la transacción han de llevarse a cabo en un tiempo inferior a 80 horas. |

## A.2 Monax

### A.2.1. Requisitos Funcionales

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
| RF01 | Instalación del Docker |
| RF02 | Conectado a: NAT |
| RF03 | Se debe poder llevar a cabo una transacción en un solo nodo |

### A.2.2. Requisitos NO Funcionales

|  |  |
| --- | --- |
| REQ. | DESCRIPCIÓN |
| R01 | Se utilizará la versión 0.16.0 de Monax |
| R02 | La implementación de la BlockChain y el desarrollo de la transacción han de llevarse a cabo en un tiempo inferior a 120 horas. |