Desplegament i monitoratge d'una xarxa Bitcoin

Treball final de grau

Pablo Martin Martí
Grau Enginyeria Informàtica
Consultora: Cristina Pérez Solà

Barcelona, 7 de juny de 2017

Agraïments

Han sigut moltes hores de treball a la meva habitació, on la pantalla era la meva única existència, per tant, un especial agraïment a la meva dona Anna, per la paciència, i un altre als meus pares i germà per els seus ànims.

També estic molt agraït per tota l'orientació rebuda per part de la meva consultora Cristina Pérez, sempre atenta i amb bones paraules. Gràcies.

Resum

Bitcoin, la nova moneda que es mou a través de la xarxa, implica un gran avanç a l'hora de fer qualsevol moviment de diners, però la seva naturalesa gairebé totalment virtual i la manca d'intermediaris fa que la seguretat agafi un especial protagonisme. En aquest anys que porta de funcionament, ara mateix vuit, ha tingut una forta evolució i el coneixement de la xarxa s'ha tornat de vital importància, per això és necessari l'existència d'eines que ens ajudin en aquesta comprensió.

Dins aquest treball tenim un programa que possibilita la creació d'una sèrie de nodes dins un entorn controlat amb el qual podem interactuar. Podem conèixer qualsevol de les tres xarxes que té Bitcoin i connectar els nodes entre ells segons la nostra conveniència. Així comprendre el seu funcionament sota una xarxa totalment privada o una pública amb bitcoins de prova i temps de procés reduït. Dins els entorns de proves podem crear transaccions i blocs, o veure dins la xarxa principal com s'integren amb altres nodes de la xarxa Bitcoin.

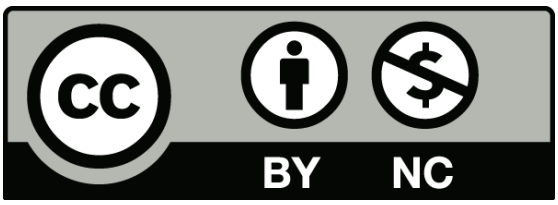
També s'afegeix una investigació per l'obtenció de diferents mètriques d'aquests nodes i extreure conclusions sobre el seu funcionament. Potents eines ajuden a crear resultats molt professionals a l'hora de presentar les dades per pantalla. Moltes són les dades que es poden extreure dels nodes, com un de sol, o com un grup. A Internet hi ha moltes gràfiques disponibles que donen dades a nivell global de la xarxa principal i de la de prova *testnet*.

Paraules clau

Bitcoin, node, xarxa pública, xarxa privada, mètriques, blocs, transacció, desplegar, blockchain, mainnet, testnet, regtest, adreces bitcoin, docker, contenidor, dockerfile.

Àrea TFG: Seguretat informàtica

Llicència



Índex

Codis

1 Introducció

1.1 Motivació

D'una manera o d'altre molta gent ja ha sentit parlar de Bitcoin, una nova moneda, que sembla que cada vegada té més impacte al món però que encara sembla lluny d'una gran quantitat de gent. A l'hora de crear aquest projecte ja són vuit anys des de la seva aparició, i la xarxa que l'alberga ja és de grans dimensions amb una gran quantitat de nodes connectats amb el protocol P2P, donant lloc a una de les més importants característiques d'aquest sistema, que és la descentralització.

Peer-to-peer (P2P)[1] és un model de comunicacions descentralitzat en el qual cada part té les mateixes capacitats i qualsevol de les parts pot iniciar una sessió de comunicació. A diferència del model client / servidor, en el qual el client realitza una petició de servei i el servidor compleix la sol·licitud, el model de xarxa P2P permet que cada node pugui funcionar com un client i com servidor a l'hora.

Els sistemes P2P es poden utilitzar per proporcionar encaminament anònim per tràfic de xarxa, entorns de computació paral·lels massius, emmagatzematge distribuït i altres funcions. La majoria dels programes P2P es centren en l'intercanvi dels mitjans de comunicació i per tant, P2P s'associa sovint amb la pirateria de programari i la violació de drets d'autor.

Aquest protocol dins Bitcoin permet als nodes fer de clients i servidors al mateix temps, i no fa necessària la participació d'un ordinador central per gestionar les diferents transaccions, imports i historial de moviments de la xarxa Bitcoin. Però no tots els nodes són iguals, ni tenen les mateixes funcions, no és el mateix fer ús d'un client Bitcoin a un ordinador sobretaula que a un mòbil, on fan ús d'un client lleuger.

Aquest projecte no va destinat a ensenyar o fomentar l'ús d'aquesta moneda, si no crear un entorn per aquelles persones que volen endinsar-se d'una manera més profunda en el seu funcionament, un espai simulat que permetrà veure el seu mecanisme intern i com respon a diferents situacions habituals dins del seu sistema. Un programa que pot ser un bon complement a la gran quantitat d'informació que hi ha a Internet.

1.2 Objectius

El conjunt de la xarxa Bitcoin, donat la seva grandària, fa difícil la seva monitorització, per això, un dels objectius és el desplegament d'una xarxa de simulació que ens permetrà la investigació dels diferents nodes i la seva interacció. Un entorn més petit on podrem observar els diferents canvis que es produiran als nodes segons si acaben de connectar-se a la xarxa, com si hi han nous blocs, transaccions o noves connexions.

I no solament hi ha una part de visualització de dades, sinó que hi ha una part d'interacció per part de l'usuari, sempre dins les xarxes de proves que posa Bitcoin a l'abast dels desenvolupadors. Tot això es posarà en marxa segons aquesta estructura:

- Investigació Bitcoin Core i client, configuració i diferents funcionalitats
- Investigació d'eina per desplegament de diferents nodes (Docker)
- Anàlisi i disseny d'una eina per desplegar nodes Bitcoin
- Desenvolupament de l'eina de desplegament
- Estudi de les diferents mètriques i dades bàsiques dels nodes.
- Anàlisi i disseny de les diferents operacions necessàries per extreure les dades de la xarxa.
- Desenvolupament de la part visual i gràfica del programari.

1.3 Metodologia

Per aconseguir desplegar una xarxa de nodes Bitcoin independent del maquinari on s'executarà el programa s'ha considerat l'opció Docker, ja que ens permet crear uns contenidors lleugers i portables que poden fer les funcions dels nodes Bitcoin i no necessiten la quantitat de recursos que per exemple necessita una màquina virtual.

Per realitzar el programari es farà ús de Python, el qual disposa les llibreries necessàries per fer la gestió dels contenidors Docker i per accedir a diferents funcions de Bitcoin. Es treballa a un entorn Linux, exactament Ubuntu 14.04 Desktop

1.4 Planificació temporal

Nombre de tarea	Duración	Comienzo	Fin
TFC	**82 días**	**mié 22/02/17**	**jue 15/06/17**
PLA DE TREBALL	14 días	mié 22/02/17	dom 12/03/17
DOCUMENTACIÓ BITCOIN	**21 días**	**lun 13/03/17**	**dom 09/04/17**
Instal·lació node complert	3 días	lun 13/03/17	mié 15/03/17
Configuració del node	3 días	jue 16/03/17	dom 19/03/17
Comandes de gestió	3 días	lun 20/03/17	mié 22/03/17
ANÀLISI EINES	1 día	jue 23/03/17	jue 23/03/17
DOCUMENTACIÓ DOCKER	**12 días**	**vie 24/03/17**	**dom 09/04/17**
Creació de contenidor	3 días	sáb 01/04/17	mar 04/04/17
Gestió de contenidors	3 días	mié 05/04/17	vie 07/04/17
Configuració xarxa contenidors	2 días	sáb 08/04/17	dom 09/04/17
IMPLEMENTACIÓ FASE 1	**30 días**	**lun 10/04/17**	**vie 19/05/17**
Programació desplegament / configuració xarxa	28 días	lun 10/04/17	mié 17/05/17
Proves i valoració opcions	3 días	mié 17/05/17	vie 19/05/17
ANÀLISI MÈTRIQUES POSIBLES	2 días	lun 01/05/17	mar 02/05/17
ESTUDI DADES I MÈTRIQUES	**9 días**	**sáb 20/05/17**	**mié 31/05/17**
Desenvolupament dades	9 días	sáb 20/05/17	mié 31/05/17
Proves i valoració	3 días	lun 29/05/17	mié 31/05/17
MEMÒRIA / PRESENTACIÓ	20 días	sáb 20/05/17	jue 15/06/17

1.5 Fitxers lliurats

El producte final del treball es compon d'aquests fitxers:

- Bitcoin.conf
- Dockerfile

- __init__.py
- nodebitcoin.py
- Main.py

El fitxer *Bitcoin.conf* porta la configuració base dels nodes, en ell es podria predefinir la xarxa d'execució del node, a qui està connectat i altres configuracions, encara que això no ens interessa d'inici, solament, en el nostre cas prepara el node per rebre comandes via RPC, indicant un usuari i contrasenya i activant l'opció *server*.

El *Dockerfile* (Annex 1) és el fitxer que construeix la imatge de Docker que serveix com referència per la creació de contenidors. Detalls importants d'aquest fitxer són l'elecció de l'entorn sota el que funcionen, el programari instal·lat, fitxer inclosos (com aquí es el *bitcoin.conf*),...

El *nodebitcoin* porta tota la lògica del programari, que és principalment la classe que representa el node i els seus mètodes per modificar-lo, o extreure'n dades. Aquí s'encapsula la informació del node i contenidor que després es farà servir per interactuar amb ell.

El *main.py* s'encarrega d'executar un petit formulari per gestionar les principals funcionalitats del programa, és molt senzill i la seva finalitat és bàsicament ajudar una mica a l'usuari a la creació de l'entorn.

2 Eines: Bitcoin i Docker

2.1 Bitcoin, conceptes bàsics

Bitcoin, com bé diu a la seva pàgina web[2] és una innovadora xarxa de pagament i una nova classe de diners, una criptodivisa sorgida al 2009 o també coneguda com moneda digital. La utilització de la tecnologia peer-to-peer és potser una de les seves característiques més rellevants, el que comporta una descentralització de la seva gestió, permetent transaccions sense cap intermediari.

Bitcoin, fa ús de la criptografia de clau pública[12], també coneguda com la criptografia asimètrica, utilitza les claus públiques i privades per verificar signatures i signar. Les claus són simplement grans nombres que s'han vinculat junts però no són idèntics (asimètrica). Una de les claus en el parell pot ser compartida amb tothom; s'anomena la clau pública. L'altra clau en el parell es manté en secret; s'anomena la clau privada. Qualsevol de les claus es poden utilitzar per xifrar un missatge; la clau oposada a la utilitzada per xifrar el missatge s'utilitza per al desxifrat.

La força del xifrat està directament relacionat amb la mida de clau, la duplicació de la longitud proporciona un augment exponencial de la força, però perjudica al rendiment. A mesura que es millora el poder de computació i es fan algoritmes de factorització més eficients, la capacitat per factoritzar nombres grans i més grans també s'incrementa.

Bitcoin crea un parell de claus que controla l'accés als bitcoins. El parell de claus consisteix en una clau privada i, derivada d'ella, una clau pública única. La clau pública s'utilitza per rebre bitcoins, i la clau privada s'utilitza per signar transaccions i poder enviar-les. Quan s'envien l'actual propietari dels bitcoins presenta la seva clau pública i una firma, i els presenta dins d'una transacció. La signatura a cada enviament és diferent, encara que totes es creen a partir de la mateixa clau privada de l'usuari.

I per fer arribar l'import a destí es fa ús de l'adreça Bitcoin, que és com una representació de la direcció d'un o varis destinataris i en la seva forma més habitual (quan correspon a un sol destinatari) representa la seva clau pública.

Cada usuari amb la intenció de fer ús de Bitcoin té un programari (Bitcoin Client) amb la funcionalitat moneder que el permet moure els bitcoins a altres destinataris, generant les transaccions.

2.1.1 Transacció

La transacció és un moviment de valor de Bitcoin i en la seva forma més bàsica es compon d'una entrada, un import i una sortida o destinatari. Les transaccions van enllaçades ja que les entrades de l'última transacció

corresponen a la sortida de transaccions anteriors, la firma d'aquestes per un remitent permet validar la nova propietat dels diners.

Les transaccions van firmades amb la clau privada del remitent, justificant la seva propietat, i utilitzen la direcció Bitcoin per assignar un destinatari. El remitent un cop preparada la transacció la fa arribar als nodes de la xarxa P2P amb la qual està connectada. I són aquest nodes els que s'encarreguen de validar les firmes criptogràfiques i el valor de les transaccions abans d'acceptar-les i enviar-les a la resta de nodes fins arribar a tota la xarxa. Aquest nodes mantenen una llista de totes les transaccions conegudes dins lo que es diu la *blockchain*.

Aquestes entrades (transaccions) que rep un usuari es mantenen indivisibles, fins l'hora de fer una despesa on es poden adjuntar per fer una sortida, o convertir-se en diverses sortides més petites. Aquest procés és transparent per l'usuari ja que és el sistema qui selecciona les entrades necessàries per fer un pagament, a més aquest procés de construcció de transaccions es pot fer fora de línia.

Una vegada s'ha creat la transacció, aquesta ha de ser transmesa a la cadena de blocs (*BlockChain*) i ser confirmada per altres nodes i acabar com part d'un nou bloc, sense importar a quin node dins la xarxa *peer-to-peer* es connecta.

2.1.2 Blocs i mineria

Els blocs són una agrupació de totes les transaccions acceptades amb el procés de mineria (mining) i els miners els fan arribar als nodes propers per que les validin i les afegeixen a la llista de blocs acceptats, i així fer la difusió a la resta de nodes.

En el procés de mineria els blocs es creen i s'omplen de transaccions no verificades que es mantenen en els nodes de la xarxa i llavors es calcula una prova de treball. Aquesta prova és com un joc molt complicat de resoldre, però a l'hora molt fàcil de comprovar que realment s'ha resolt correctament, amb el pas del temps aquest joc s'ha tornat cada vegada més complex i és necessari un maquinari cada vegada més potent i especialitzat.

Dit d'altre manera, els miners creen un hash del bloc amb les transaccions no confirmades, seguint una sèrie de premisses (que dificulten la creació d'aquest hash), i així assegurar la seva integritat, aquest hash es col·loca al final del bloc, i a més s'afegeix el hash del bloc anterior per fer-hi com un enllaç entre ells.

La mineria és important per crear nous bitcoins, i perquè genera confiança a l'assegurar que les transaccions només es confirmen si s'ha dedicat suficient potència computacional al bloc que les conté. Més blocs signifiquen més computació, el que significa més confiança.

Tot aquest procés genera una recompensa per al miner, per tot l'esforç de contribució a la xarxa de Bitcoin, aquest import surt de cada transacció que es fa, i és important quan es treballa amb les xarxes de prova de Bitcoin.

Actualment els blocs ocupen aproximadament 1Mb i porten un número variable de transaccions, aquesta informació es pot veure al dia amb un explorador de la cadena de blocs com Blockchain.info[11].

2.2 Xarxa Bitcoin

La xarxa de Bitcoin[6] està composta de nodes connectats entre ells a través del protocol P2P, aquest sistema permet que tots el nodes siguin iguals entre ells i que no faci falta un servidor central, que és un dels grans avantatges de Bitcoin. Els nodes s'encarreguen de validar i propagar les transaccions i els blocs. Quan un node nou es connecta a la xarxa Bitcoin aquest fa un rastreig per trobar un node que ja pertany a la xarxa i quan el localitza fa una connexió TCP a través del port 8333 (*mainnet*) i llavors l'envia un missatge per donar-se a conèixer (*handshake*). Aquest missatge conté informació com el protocol, els serveis locals, l'hora actual, la IP del node remot i del local, la versió del programari i el bloc més alt del que disposa. Si el node destí l'accepta confirma la connexió (*verack*) i pot enviar un missatge de versió.

Per trobar aquest primer node, es pot fer a través de DNS, on s'utilitza una llista de servidors llavor, aquests servidors DNS retornen les adreces IP dels nodes complets a la xarxa Bitcoin per ajudar en el descobriment d'homòlegs. Aquesta és l' opció per defecte dels clients Bitcoin (*-dnsseed=1*). També es pot determinar manualment la IP d'una llavor, que després d'aconseguir la direcció d'altres llavors es desconnectarà d'aquesta (*-seednode*). Hi ha un altra opció manual que permet la connexió directa amb un o varis nodes (*-connect*). Una vegada ja s'ha connectat a un primer node, i el demana la seva llista de companys per connectar-se, el node crea el fitxer "peers.dat" on guardarà la informació dels parells per facilitar-ne posteriors reconnexions.

Una vegada establerta la connexió enviarà un missatge *addr* on inclourà la seva IP perquè els nodes la propaguin a altres nodes, a més també podrà fer ús del missatge *getaddr* per demanar la llista d'altres nodes a qui anunciar-se. És important que els nodes mantinguin diverses connexions a diferents nodes, ja que la xarxa canvia i alguns nodes desapareixen i altres són nous.

2.3 Node Bitcoin

Els nodes formen la xarxa bitcoin, i aquests poden ser de diferents tipus segons la seva funcionalitat. La principal, la qual tenen tots, és la d'enca-

minament que permet la validació dels blocs i transaccions, apart de la seva propagació a altres nodes.

Els nodes complerts són aquells que tenen tota la base de dades de la cadena de blocs i verifiquen de forma autònoma qualsevol transacció. Els nodes que no tenen tota la base de dades també poden validar però amb un altre mètode anomenat SPV o pagament simplificat.

Els nodes miners s'encarreguen de la creació de nous blocs, amb la resolució d'un algorisme de prova de treball. Aquest nodes tenen una còpia complerta de la base de dades.

La funció moneder és aquella que permet fer i rebre pagaments. Molts dispositius mòbils i amb recursos limitats es diuen clients lleugers quan no tenen la base de dades complerta de la cadena de blocs.

Un altre tipus de nodes són els "pruned", o nodes podats[7], que van sorgir com a solució per evitar l'emmagatzematge de la base de dades complerta, que cada vegada requereix més capacitat d'espai. Aquest nodes no tenen tota la base de dades de blocs sencera, però també permeten fer validacions complertes de les transaccions. Per evitar qualsevol possible situació de doble despesa descarreguen la part que manca i així garanteixen una validació correcta.

2.4 Xarxes d'execució Bitcoin

Bitcoin pot operar en tres tipus de xarxes, la principal (*Mainnet*), que és la xarxa real de Bitcoin, i després dues xarxes més que estan pensades per desenvolupadors, *testnet* i *regtest*.

Testnet: és una xarxa de blocs alternativa per fer proves, no utilitza bitcoins reals i tots els processos són més ràpids que a la xarxa principal. Es poden fer totes les funcions de la xarxa real i té una nomenclatura diferent de les direccions per evitar confusions. Hi ha ja tres versions d'aquesta xarxa en les que principalment s'ha modificat el bloc gènesis. Les diferencies amb la xarxa principal:

- El port d'escolta és 18333 (en lloc del 8333)
- El port de connexió RPC per defecte és 18332 (en lloc de 8332)
- Bootstrapping (Elecció de parells de la base de dades local segons la data de descobriment i amb una mica d'aleatorietat) utilitza diferents llavors de DNS.
- Un valor diferent de camp ADDRESSVERSION assegura que cap adreça d'aquesta xarxa treballarà a la xarxa de producció. (0x6F en lloc de 0x00)

- Els bytes de capçalera del missatge de protocol són 0x0B110907 (en lloc de 0xF9BEB4D9)
- La dificultat de creació de blocs és la meitat, i encara que s'incrementa, com passa a la xarxa principal quan es creen nous blocs, es restablirà si no hi troba un després de 20 minuts, però sol per un bloc, després recupera el seu valor anterior.
- Un bloc gènesi especificat per aquesta xarxa
- Es pot experimentar amb transaccions no estàndard.

Regtest: Quan va sortir la versió 0.9.0 de Bitcoin Core[8], en març del 2014, es va afegir aquesta xarxa que també serveix per fer proves com l'anterior però que aquesta crea una cadena de blocs privada, on l'usuari pot escollir quan crear nous blocs, obtenint un control absolut sobre la xarxa. A més és molt més lleuger que la Testnet i la creació de blocs és quasi instantània.

Per aprofitar aquesta xarxa hem de conèixer el funcionament de Bitcoin respecte al sistema de recompenses que s'obté per la mineria[6]. Cada nou bloc implica guanyar un import estipulat, actualment en la xarxa principal de Bitcoin és de 12.5 bitcoins, però en la Regtest, com és una xarxa nova i privada és el valor antic de 50 bitcoins.

No serà possible la despesa d'aquest import fins tenir 100 blocs confirmant la transacció, o sigui, si vam crear un primer bloc a la Regtest i vam guanyar 50 bitcoins, per poder gastar-los necessitem crear 100 blocs més. Cada 210.000 blocs aquest valor que es guanya, es redueix a la meitat. El port d'escolta en aquest cas és el 18444.

2.5 Configuració Bitcoin

Hi ha moltes opcions disponibles en la configuració dels nodes Bitcoin[13], es destaquen algunes que ens seran necessàries en la creació de l'entorn. L'arxiu bitcoin.conf és el que per defecte porta la configuració, encara que es pot canviar al igual que la seva ubicació.

-rpcuser=usuari: Assignar usuari per activar les comandes RPC.

-rpcpassword=password: Assignar una contrasenya per activar les comandes RPC.

-addnode=≪IP≫: Per afegir nodes de confiança amb els quals connectar.

-connect=≪IP≫: Per indicar nodes concrets als que connectar-se, solament es connectarà als indicats amb aquesta opció, ignorant qualsevol altre encara que estigui a la mateixa xarxa local.

-testnet=1: Activació del node dins la xarxa de prova *testnet*.

-regtest=1: Activació del node dins la xarxa de prova *regtest*.

-prune=<n>: Permet reduir els requisits d'emmagatzematge, permetent la poda (eliminació) dels blocs més antics. Aquesta configuració requereix tornar a descarregar tota la cadena de blocs. (Per defecte: 0 = Desactivat, 1 = permet la poda manual a través d'RPC, >550 = automàticament poda els arxius de bloc per mantenir-se sota la mida destí indicada en MiB.)

-server=1: Activa la possibilitat d'ús de les comandes JSON-RPC.

2.6 Docker

Docker[3] és un projecte de codi obert que permet desplegar aplicacions dins d'uns contenidors de programari, que a diferència de les màquines virtuals no té la necessitat de crear un nou sistema operatiu complert, sinó que es basa en aprofitar els recursos de la màquina arrel per crear espais independents privats, amb els seus propis recursos, serveis, sistema d'arxius, etc.

Això fa que siguin eficients, lleugers, autònoms i que garanteixin que el programari sempre funcionarà igual, independentment d'on es vulgui desplegar. Aquest programa l'utilitzarem per crear l'entorn Bitcoin, ja que cada contenidor portarà a terme la funció d'un node de la xarxa, lo que ens permetrà fer diferents proves i el seguiment d'aquests.

Els contenidors es creen a partir d'una imatge, que és un paquet executable, lleuger, autònom que inclou tot el necessari per arrancar una peça de programari, amb el codi, llibreries, variables d'entorn i fitxers de configuració. Llavors un contenidor és una instància en temps d'execució d'una imatge. S'executa, per defecte, completament aïllat de l'entorn d'acollida, només s'accedeix a arxius host i ports si està configurat per a fer-ho.

Docker pot construir imatges de forma automàtica mitjançant la lectura de les instruccions d'un Dockerfile[5]. Un Dockerfile és un document de text que conté totes les comandes necessàries per un usuari per muntar una imatge.

2.6.1 Creació d'un contenidor

Les operacions descrites aquí són considerant un entorn Linux, amb terminal. Per crear un contenidor amb docker s'utilitza el paràmetre *run*:

Codi 1: Execució d'un contenidor

```
docker run [OPTIONS] IMAGE [COMMAND] [ARG...]
```

Entre les diferents opcions disponibles a l'hora d'executar un contenidor destaquem, l'execució en segon pla, assignar un nom de host, un nom descriptiu, personalitzar els DNS, etc. Després es posa el nom de la imatge base de la qual es crea la instància i després opcionalment executar un comanda dins ja del contenidor amb els seus arguments.

Codi 2: Opcions execució d'un contenidor

```
#Execució d'un contenidor en segon pla.
docker run -d nom_imatge
#Execució d'un contenidor amb nom descriptiu node_bitcoin
#i amb l'execució d'una comanda.
docker run --name some_bitcoin nom_imatge bitcoind
```

2.6.2 La xarxa Docker

Després de la instal·lació de Docker ens apareix una nova interfície de xarxa amb el nom de *docker0*, aquesta porta la configuració base de tots el futurs contenidors, sempre que utilitzem la xarxa per defecte. Les xarxes que porta Docker d'un inici serien:

bridge Aquesta és la xarxa per defecte al menys que s'utilitzi l'opció –*network* per especificar-ne un altra, representa la *docker0*.

none Aquesta no porta interfície de xarxa.

host Trenca l'aïllament contenidor-arrel.

A més, Docker, permet crear les teves xarxes i així controlar quins contenidors es connecten entre ells, també per habilitar la resolució automàtica de DNS del noms dels container a una adreça ip, etc.

Si creem diversos contenidors sense canviar la xarxa per defecte tots ells serien part d'una mateixa xarxa interna, amb connexió a Internet segons si l'ordinador arrel té.

3 Anàlisi previ desenvolupament

3.1 Estat d'art

Amb els anys que porta Bitcoin en marxa és difícil pensar que no hagi tot tipus de programes vinculats a la xarxa que permetin tot tipus d'operacions. Però encara que si que hi ha molt programari no sembla que hagin grans solucions de desplegament.

Una vegada conegudes les eines principals amb que treballar, era necessari un llenguatge de programació intuïtiu per començar a desenvolupar el programari, i pensant que treballaríem amb contenidors Docker, que representarien els diferents nodes a desplegar amb aquesta eina, necessitaríem un llenguatge amb una biblioteca ja consolidada com Docker SDK for Python[10], que és la que recomana a la pròpia pàgina de Docker.

L'alternativa de gak[20] ens permet desplegar a la xarxa *regtest* dos nodes i fer petites proves sobre ells, aquesta també funciona sobre Docker, i funciona amb comandes RPC, algunes de elles simplificades amb àlies creats al Dockerfile. Pots crear noves adreces, generar blocs i enviar pagaments.

Una altra opció és de freewil[21], molt similar a l'anterior, dos nodes dins la xarxa de proves privada, amb la gestió via comandes, també utilitzant els contenidors Docker i els àlies per facilitar l'entrada de comandes. Respecte les funcionalitats són les mateixes, creació d'adreces, enviament de bitcoins, generació de blocs, posar tot d'inici, etc. Les solucions trobades són bastant senzilles, es limiten a la xarxa privada i a dos nodes únicament, però donen un mínim de sortida per fer un programa de característiques semblants.

3.2 Desplegament

Amb la llibreria Docker per python[10] es facilita molt la creació, aturada, posada en marxa i eliminació dels containers, a més de poder executar diferents instruccions sota el seu entorn. Això ens ajudaria a crear una xarxa segons les nostres necessitats i amb possibilitat d'adaptar-la sense tenir que començar des de el principi.

Ara que ja coneixia com desplegar l'entorn on funcionarien els nodes, era necessari que aquests contenidors tinguessin tot el necessari per llançar un node bitcoin. Un primer anàlisi hem va portar a instal·lar tots els requisits necessaris[6] per llançar Bitcoin en un entorn propi, però els errors van ser diversos, sense arribar a una conclusió clara, per tant, donat que actualment ja hi ha entorns creats i en funcionament, vaig inclinar-me per aquesta opció.

A més no feia falta instal·lar un entorn per desenvolupadors, ja que no volia fer canvis sobre el codi principal, per tant, era necessari una versió

binària que permetés totes les funcionalitats descrites per als nodes. Necessitava llavors una imatge Docker que tingués tot l'entorn Bitcoin instal·lat i llest per posar en marxa.

Realment les opcions trobades creen un entorn amb un node Bitcoin funcionant, però cap s'adaptava totalment a la necessitat d'aquest projecte, i és que el propòsit d'aquest és poder desplegar diferents tipus de nodes bitcoin, i a més que permetés l'enviament de comandes via RPC. Amb això últim s'aconseguiria poder enviar les diferents configuracions als nodes, tant per seleccionar la xarxa d'execució, com per indicar els nodes amb qui són connectats, etc.

Sobre les versions trobades, la de kylemanna[14] no era molt actualitzada i encara que feia ús de l'ultima versió de Bitcoin no era el que hem proposava, donat que les actualitzacions podien variar algunes de les funcionalitats del meu programa. L'opció seegno[15] tenia diferents versions a disposar, però tampoc hi era massa actualitzat i es quedava a la versió 0.13 de bitcoin.

També hi havia l'opció de cdecker[16], però aquesta estava més pensada per desenvolupadors, i feia tot el procés de compilació innecessari en aquest projecte. I finalment l'opció de amacneil[9], que si tenia diverses versions de bitcoin i que estava molt actualitzat, però no tenia activat les comandes per RPC.

Aquesta última opció, encara que no idònia, serviria per aconseguir els contenidors amb un node bitcoin instal·lat, utilitzaria l'última versió en aquest moment (0.14.0) i faltaria activar les comandes RPC modificant i copiant el fitxer de configuració Bitcoin, i treien l'execució de la versió servidor de bitcoin (bitcoind) per poder executar segons uns paràmetres concrets.

3.2.1 Connexions

Ara que ja tenia clar com desplegar diferents nodes, i com funcionava la xarxa Docker per defecte, coneixia que els nodes es podien veure entre ells, perquè pertanyien a una mateixa xarxa local. Llavors la primera consideració a tenir era els diferents comportament dels nodes dins les tres xarxes d'execució Bitcoin.

En les tres xarxes, sense cap restricció, els nodes comencen a fer la cerca d'altres nodes per descarregar-se la cadena de blocs, la que correspon a cada xarxa. Per disposar d'informació actualitzada sobre el nombre de blocs, en el cas de la mainnet tenim la web de blockchain[11], i en el cas de la testnet tenim la blockr[17].

La regtest crea una xarxa privada solament amb un bloc gènesis, per tant, sense cap interacció nostre no té cap moviment, per tant, seria important permetre simular pagaments i creació de blocs (mineria) per veure com funcionen dins d'aquesta.

Llavors, per el tema de les connexions, per aconseguir una comunicació concreta entre els nodes, vaig pensar primer a nivell Docker, contenidors amb diferents configuracions de xarxa, sense connexió a internet gestionant les DNS o inclús fent ús del tallafoc per defecte de Linux (Iptables) per aconseguir crear algunes limitacions.

Altre opció era a nivell de Bitcoin, ja que la configuració d'aquest també permet afegir o limitar les connexions a un o varis nodes. Per la seva poca complexitat aquesta era la millor opció. Treballaria amb l'opció (*-connect*) per limitar les connexions a uns nodes concrets.

3.2.2 Interacció nodes

Amb els nodes desplegats i la connexió entre ells, ara tocava la interacció aprofitant les xarxes de prova. La primera, i més bàsica és fer transaccions entre nodes, moure quantitats de bitcoins entre ells i potser, després, portar un control de les transaccions realitzades i quan es confirmen a cada node.

Això es porta a una segona acció, que és la creació de blocs, a més de permetre confirmar les transaccions i així permetre fer més despeses , ens permet guanyar bitcoins amb el sistema de recompenses. En el cas de la *regtest* aquest procés és fàcil i ràpid, però no així a la *testnet*.

La *testnet* és una xarxa de prova molt semblant a la principal però que redueix els temps per minar un bloc, i funciona amb adreces i bitcoins diferents. Això implica principalment que els costos de computació a l'hora de minar són més alts i potser mala idea utilitzar aquesta opció per aconseguir bitcoins.

Per solucionar aquest problema s'han creat les *faucets*[22] (traduït: aixetes), que són pàgines que t'envien bitcoins per aquesta xarxa perquè puguis fes proves, amb la única condició que els retornis quan hagis acabat.

3.3 Mètriques

En un principi, abans de parlar de mètriques tenia que veure com extreure les dades del nodes, i Bitcoin t'ho posa fàcil al donar-te les dades en JavaScript Object Notation (JSON), un format molt llegible i tractable. Una vegada tindria les dades havia que valorar quines d'aquestes serien interessants de tractar, ja que cada xarxa d'execució Bitcoin, per el seu diferent funcionament, tenen diferents mètriques que encaixen.

A la xarxa principal (*mainnet*), no hauria cap interacció per part de l'usuari, per tant, era especialment interessant per veure la propagació de la cadena de blocs en nodes nous o interns, aquells que depenen d'un altre node de la xarxa per descarregar-se els blocs.

En el cas de la *testnet*, es coneix que és pública i funciona igual que la principal amb bitcoins, adreces i blocs de prova, i que també té reduïts els temps dels processos. En aquest cas si podem interactuar, però els valors de temps no són vàlids per conèixer la realitat del funcionament de la xarxa Bitcoin. I finalment la *regtest* en que els temps són reduïts a la mínima expressió, però en aquesta xarxa, que si és privada, si podem obtenir valors globals del conjunt dels nostres nodes.

A l'hora de pensar en les mètriques vaig agafar com a primera referència la pàgina web de blockchain[11], aquesta web té un apartat d'estadístiques i gràfiques, però té moltes dades genèriques que no s'adaptaven a la nostra xarxa, té informació com el preu de mercat, mida de la blockchain, bitcoins en circulació...

Continuant amb la cerca de noves webs, ens trobem amb la de statoshi[19], aquesta té una gran varietat de categories i panells donant tot tipus de informació sobre la xarxa Bitcoin, no solament això, gràcies a la seva eina *Grafana* ens dona un entorn visual per crear les nostres gràfiques. Però per fer ús d'aquesta eina, darrera té que haver-hi un programa que s'encarregui de guardar les dades que nosaltres volem extreure als nostres nodes, i *Grafana* ens dona diverses possibilitat, sent una de les oficials, *Graphite*[18].

Aquest programa, encara que també pot fer gràfiques sembla no tan visual i potent com *Grafana*, i solament en aquest cas es faria servir com base de dades. Però ens pot servir com pas intermedi, per tant, tenia que veure com passar les dades a *Graphite*.

Com estem treballant amb contenidors Docker una opció és crear-ne un amb el *graphite* llest per enviar les dades, com aquesta solució[18], amb la qual solament tindríem que enviar les dades des de el programa a aquest contenidor dins la mateixa xarxa.

Carbon, és el servei o grups de serveis que s'encarreguen de constituir la base de dades de *Graphite*, és ell qui té que rebre les dades del nostre programa segons una de les opcions disponibles segons el cas.

4 Desenvolupament

Hem vist de la necessitat d'utilitzar Docker[3] per aconseguir desplegar una sèrie de nodes per fer proves, per tant, és necessari la instal·lació del programa que té versions per Mac, Windows i Linux. Les proves s'han fet sobre un sistema operatiu Linux Ubuntu Desktop 14.04 i per tant detallaré el procés dins aquest entorn.

Les comandes necessàries es basen en la versió "Community Edition" de Docker, i ja compleix els requeriments per fer el desplegament. Després

d'instal·lar veiem que tota execució del programa s'ha de fer amb permisos d'administrador, lo que és inviable a l'hora de treballar amb el programa de desplegament. La pàgina web d'ajuda del propi Docker[4] ens dona les pautes a seguir per evitar aquesta situació i així el nostre programa podrà executar les instruccions sense problemes.

4.1 Desplegament

Una vegada ja tenim aquests prerequisits el primer pas al programari és carregar el Dockerfile (veure Annex 1) que ens permetrà tenir una imatge preparada per la creació del nodes Bitcoin.

Codi 3: Python, construcció d'una imatge Docker

```
#Creació d'una imatge Docker que és la base per els contenidors
client.images.build(path='.', tag='node_bitcoin', forcerm=True)
```

Indiquem un punt al *path* per indicar que el fitxer és a la mateixa carpeta on hi ha el programa, li posem el nom de 'node_bitcoin' a la imatge i amb l'última opció forcem l'esborrament d'uns contenidors intermedis que Docker utilitza per la creació de la imatge.

Com que els nodes són el concepte principal amb el que treballarem, crearem una classe per representar-los, i les dades que demanarem d'inici serà la xarxa a la que volem que pertany i als nodes amb els quals vol estar connectats.

Codi 4: Python, classe node bitcoin

```
class node_bitcoin:
    def __init__(self, network, node=[]):
```

Quan es crea una nova instància de la classe aquesta executa un nou container amb un node_bitcoin inclòs i funcionant.

Codi 5: Python, creació d'un contenidor

```
#Creació d'un contenidor amb Bitcoin sense especificar connexions
self.client = client.containers.run(image='node_bitcoin', command=
    'bitcoind␣'+self.net, detach=True)
```

A l'hora d'utilitzar la comanda s'ha d'indicar la imatge en la qual es basa, utilitzar l'opció *detach* per activar la càrrega en segon pla i executar el servei de Bitcoin amb les seves diferents opcions, que en el nostre cas és a la xarxa que pertany i si està connectat a uns nodes concrets.

Un altre detall important és que cada instància guardarà l'objecte contenidor creat, a més de la xarxa bitcoin a la que pertany i dels nodes als que hi és connectat.

En el cas de les connexions es consideren 3 situacions, una primera on el node no té cap restricció de connexió, per tant, es pot connectar a qualsevol node vàlid per aconseguir els blocs de la xarxa. En aquest cas no implica incloure res a l'hora de l'execució del servei Bitcoin.

Un segon cas on el node es connecta a un altre extern, que no pertany a la nostra xarxa privada, i que és el propi programa qui el proporciona. Encara i així, si tinguéssim la necessitat d'obtenir direccions de nodes de la xarxa principal podem fer ús de l'explorador blockchain[11].

I finalment, el tercer cas que es connecta a un o varis nodes dins el nostre entorn tancat, per exemple per construir una xarxa que depèn d'un sol node per arribar a la xarxa externa de Bitcoin, sent aquest l'encarregat de la propagació dins la xarxa interna.

Codi 6: Python, creació d'un contenidor amb connexió

```
#Creació del contenidor amb Bitcoin especificant nodes amb els
    quals connectar
#node1
client.containers.run(image='node_bitcoin', command='bitcoind␣
    {connect=node2)
#node2
client.containers.run(image='node_bitcoin',␣command='bitcoind
    {connect=node3)
#node3
client.containers.run(image='node_bitcoin', command='bitcoind␣
    {connect=node4)
#node4
client.containers.run(image='node_bitcoin',␣command='bitcoind
    {connect=nodeExtern)
# Representació:
#Node 1 - Node 2 - Node 3 - Node 4 - Node extern
```

4.2 Interacció nodes

Una vegada ja tenim desplegada la xarxa necessitem els mètodes per poder fer interaccions amb els nodes que la formen, o inclús canviar les seves connexions. Això ens permetrà treure´n més profit a les xarxes de proves.

La generació de blocs ens ajudarà a aconseguir recompenses de bitcoins, i confirmar les transaccions per poder fer-les servir, cal anar amb compte que encara que la xarxa de prova *testnet* redueix el temps i dificultat d'aquest procés, encara segueix sent pesat i llarg per un ordinador particular.

Cada cert temps aquesta xarxa de prova es regenera i es deixa amb solament el bloc gènesis (el primer bloc), però a l'hora d'escriure aquestes línies porta una gran quantitat de blocs i la seva dificultat ha anat incrementant. Els blocs són de menys pes que els de la xarxa principal, per tant, la descàrrega de la cadena de blocs és més ràpida.

Codi 7: Python, generació de blocs

```
def miningBlock(self, value):
#Generació de blocs
#value, nombre de blocs a generar
return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣generate
    ␣'+str(value))
```

La creació d'adreces va pensada especialment per quan es necessita una per fer un enviament de diners, per tant, aquest mètode l'he associat amb el d'enviament de bitcoins.

Codi 8: Python, creació d'una nova adreça

```
def getNewAddress(self):
        #Creacio d'una nova adreça
        return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣
            getnewaddress')
```

Al codi podem veure com utilitza el mètode anterior de creació d'adreces. Aquest mètode espera dos valors, el valor de l'import a enviar, i el nombre del node Bitcoin. Llavors aconsegueix una adreça del node destí i li envia la quantitat introduïda.

Codi 9: Python, enviament de bitcoins

```
def sendToAddress(self, value, node):
        #Enviament de bitcoins a un node desti
        address = nodes[node-1].getNewAddress()
        return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣
            sendtoaddress␣'+address+'␣'+str(value))
```

Per no tornar a crear la xarxa d'un inici, aquest mètode ens proporciona la possibilitat d'afegir o esborrar una connexió, així ens permet fer canvis en l'enllaç dels diferents nodes per estudiar diferents comportaments. El valor que es demana en la creació d'una connexió és la instància del node destí.

Codi 10: Python, creació i eliminació d'una connexió

```
def addConnection(self, node):
        #Afegir una connexio a un node desti
        addressIP = node.getIP()
        return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣
            addnode␣'+addressIP+':'+self.port+'␣onetry')
    def delConnection(self, node):
        #Eliminar una connexio d'un node'
        addressIP = node.getIP()
        return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣
            addnode␣'+addressIP+':'+self.port+'␣remove')
```

4.3 Dades

Una vegada desplegada la xarxa, el següent pas era començar a extreure dades d'aquests nodes, per controlar la seva situació en un moment concret. Aquestes dades venien amb format JSON, que amb un procés de descodificació podíem referenciar-la amb facilitat.

En aquest primer cas tenim l'import del moneder disponible per fer despesa, aquest valor surt de les transaccions rebudes i ja confirmades, una sola instrucció ja ens retorna aquest valor sense necessitat de fer el procés de descodificació.

Codi 11: Python, extreure import dins el moneder

```
def getBalance(self):
        #Retorna el valor de bitcoins disponibles per fer despesa
        return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣
            getbalance')
```

Una altra dada important són els blocs descarregats en un moment concret, per conèixer com evoluciona la sincronització amb la cadena de blocs, sobretot si tens un node que depèn directament d'un altre de la xarxa interna.

Codi 12: Python, extreure nombre de blocs descarregats

```
def getBlocks(self):
        #Retorna el numero de blocs descarregat de la xarxa
        return self.client.exec_run(cmd='bitcoin-cli␣'+self.net+'␣
            getblockcount')
```

Per confirmar les connexions realitzades, o en el cas que hem escollit un node lliure per connectar-se a altres nodes, no pot faltar la llista IPs d'aquests.

Codi 13: Python, llista dels nodes connectats

```
def showAddressConnect(self):
        #Mostra el nodes amb qui s'esta connectat'
        listAdd = []
        listAddresses = decode(self.client.exec_run(cmd='bitcoin-
            cli␣'+self.net+'␣getpeerinfo'))
        for i in listAddresses:
            listAdd.append(i['addr'])
        return listAdd
```

5 Conclusió

En aquest treball hem creat una eina que ens permet desplegar una sèrie de nodes bitcoin en funcionament, permeten escollir la xarxa d'execució sota la que treballar. Amb això s'ha aconseguit comprovar els diferents comportaments dels nodes.

S'ha fet un estudi intensiu dels conceptes bàsics de Bitcoin, les diferents xarxes, nodes, funcionalitats, tot per construir i entendre els diferents comportaments del nodes a les xarxes d'execució, i a més com cercar el màxim aprofitament d'aquestes.

En les proves realitzades dins la xarxa *mainnet* hem vist la diferència de connectar amb un node extern i un de local, que encara que connectats, ha sigut imprescindible posar la xarxa dins de la llista blanca perquè comences a descarregar-se els blocs. O sigui, si un primer node estava connectat a un node extern, de seguida començava a descarregar la cadena de blocs, però, perquè un segon node connectat solament al primer es descarregués la cadena, aquest el tindria que tenir dins la seva xarxa de confiança. Això també s'aplica a la xarxa *testnet*. Però encara que s'ha aconseguit la connexió a un node local, no per un tercer node connectat al segon, no permet la connexió directa a aquest node.

El projecte ha sigut una experiència molt interesant, encara que ha quedat molt de camí per recórrer. Però tota la investigació sobre el funcionament de Bitcoin i les diferents eines per treure´n més profit ha sigut molt valuosa.

Però per un futur caldria finalitzar la integració amb algun programa de control de mètriques, seguint la investigació de *Grafana*, *Graphite* i *Carbon*, també fer un panell per la gestió dels diferents nodes del desplegament i ampliar les possibilitats de connexió dels nodes si fos possible.

6 Glossari

Protocol P2P: Permet a una xarxa d'ordinadors funcionar sense clients ni servidors fixos, sinó que són una sèrie de nodes que es comporten com iguals entre si.
Model client / servidor: es recolza en terminals (clients) connectats a un ordinador que els proveeix d'un recurs (servidor). D'aquesta manera els clients són els elements que necessiten serveis del recurs i el servidor és l'entitat que posseeixen el recurs.
Encaminament anònim: És una tècnica per fer que la identitat i la ubicació de la informació dels locutors d'una comunicació és anònima. **Client Bitcoin:** També es diu moneder, és un programari que ens permet enviar i rebre bitcoins.
Client lleuger: És un tipus de programari Bitcoin que no funciona com un node de xarxa P2P i no descarrega la blockchain sencera.
Bitcoin Core: és el codi font de Bitcoin i és la implementació de referència de la xarxa Bitcoin.
Docker: és un projecte de codi obert que automatitza el desplegament d'aplicacions dins de contenidors de programari, proporcionant una capa addicional d'abstracció i automatització.
Criptografia de clau pública: També coneguda com xifrat asimètric. Un missatge xifrat amb la clau pública d'un destinatari no pot ser desxifrat per ningú (incloent al que el va xifrar), excepte un posseïdor de la clau privada corresponent, en teoria el seu propietari i la persona associada amb la clau pública utilitzada.
Criptodivisa: És un mitjà digital d'intercanvi.
Hash: Els hash o funcions de resum són algoritmes que aconsegueixen crear a partir d'una entrada (ja sigui un text, una contrasenya o un arxiu, per exemple) una sortida alfanumèrica de longitud normalment fixa que representa un resum de tota la informació que se li ha donat.
Host: O també dit amfitrió, es fa servir per referir-se a les computadores o altres dispositius connectats a una xarxa que proveeixen i utilitzen serveis d'aquesta.

7 Bibliografia

Referències

[1] Wikipedia P2P - https://es.wikipedia.org/wiki/Peer-to-peer

[2] Pàgina oficial de Bitcoin http://www.bitcoin.org

[3] Pàgina oficial de Docker, https://www.docker.com/

[4] Post-installation steps for Linux
https://docs.docker.com/engine/installation/linux/linux-postinstall/

[5] Dockerfile reference
https://docs.docker.com/engine/reference/builder/

[6] Mastering Bitcoin, Andreas M. Antonopoulos, O'REILLY Atlas
http://chimera.labs.oreilly.com/books/1234000001802

[7] Pro's and Con's on Bitcoin Block Pruning
https://news.bitcoin.com/pros-and-cons-on-bitcoin-block-pruning/

[8] ¿Qué es la Bitcoin Testnet y la Bitcoin Regtest?
https://www.oroyfinanzas.com/2015/02/que-bitcoin-testnet-regtest/

[9] https://github.com/amacneil/docker-bitcoin

[10] Docker SDK for Python
https://docker-py.readthedocs.io/en/stable/index.html

[11] Explorador de bloques de Bitcoin
https://blockchain.info/

[12] Asymmetric-Key Cryptography
https://www.cs.cornell.edu/courses/cs5430/2013sp/TL04.asymmetric.html

[13] Bitcoinwiki - Running Bitcoin
https://en.bitcoin.it/wiki/Running_Bitcoin

[14] kylemanna/docker-bitcoind
https://github.com/kylemanna/docker-bitcoind

[15] seegno/bitcoind
https://github.com/seegno/docker-bitcoind

[16] cdecker/docker-bitcoin-dev
https://github.com/cdecker/docker-bitcoin-dev

[17] Testnet BTC Block Explorer
http://tbtc.blockr.io/

[18] Docker Image for Graphite & Statsd
https://github.com/hopsoft/docker-graphite-statsd

[19] Statoshi.info - Realtime Bitcoin Node Stats
http://statoshi.info/

[20] gak/docker-bitcoin-regtest
https://github.com/gak/docker-bitcoin-regtest

[21] freewil/bitcoin-testnet-box
https://github.com/freewil/bitcoin-testnet-box

[22] Tesnet - https://en.bitcoin.it/wiki/Testnet

8 Annex

8.1 Dockerfile

És un fitxer que serveix per la construcció d'imatges a Docker, i aquestes imatges són la base per la creació de contenidors. En el cas del programa d'aquest treball s'ha utilitzat un Dockerfile existent[9] com referència per després acabar d'ajustar-ho a les nostres necessitats.

Les últimes dues línies copien el fitxer de configuració Bitcoin adaptat per aconseguir realitzar comandes RPC, a més s'ha tret l'execució del Bitcoin Core que portava el Dockerfile original, ja que així, després d'un nou contenidor l'executem amb la configuració sol·licitada per l'usuari.

Algunes instruccions destacades dintre aquest fitxer:

FROM estableix la imatge de base per a instruccions posteriors.

RUN Executarà una comanda en una nova capa superior de la imatge actual que compromet resultats posteriors.

ENV Estableix una variable d'entorn amb un valor. Aquest valor estarà disponible en la resta del fitxer.

EXPOSE informa els ports que el contenidor escolta en temps d'execució.

VOLUME Crea un punt de muntatge amb el nom especificat i el marca per ser accedit des de l'ordinador arrel o altre contenidor.

Codi 14: Dockerfile

```
FROM debian:jessie

RUN groupadd -r bitcoin && useradd -r -m -g bitcoin bitcoin

ENV BITCOIN_VERSION 0.14.0
ENV BITCOIN_URL https://bitcoin.org/bin/bitcoin-core-0.14.0/
    bitcoin-0.14.0-x86_64-linux-gnu.tar.gz
ENV BITCOIN_SHA256 06
    e6ceeb687e784e9aaad45e9407c7eed5f7e9c9bbe44083179287f54f0f9f2b
ENV BITCOIN_ASC_URL https://bitcoin.org/bin/bitcoin-core-0.14.0/
    SHA256SUMS.asc
ENV BITCOIN_PGP_KEY 01EA5486DE18A882D4C2684590C8019E36C2E964

RUN set -ex \
        && apt-get update \
        && apt-get install -y --no-install-recommends ca-
            certificates wget \
        && rm -rf /var/lib/apt/lists/* \

        # grab gosu for easy step-down from root
```

```
        && gpg --keyserver pool.sks-keyservers.net --recv-keys
            B42F6819007F00F88E364FD4036A9C25BF357DD4 \
        && wget -qO /usr/local/bin/gosu "https://github.com/tianon
            /gosu/releases/download/1.7/gosu-$(dpkg␣--print-
            architecture)" \
        && wget -qO /usr/local/bin/gosu.asc "https://github.com/
            tianon/gosu/releases/download/1.7/gosu-$(dpkg␣--print-
            architecture).asc" \
        && gpg --verify /usr/local/bin/gosu.asc \
        && rm /usr/local/bin/gosu.asc \
        && chmod +x /usr/local/bin/gosu \

        # install bitcoin binaries
        && BITCOIN_DIST=$(basename $BITCOIN_URL) \
        && wget -O $BITCOIN_DIST $BITCOIN_URL \
        && echo "$BITCOIN_SHA256␣$BITCOIN_DIST" | sha256sum -c - \
        && gpg --keyserver pool.sks-keyservers.net --recv-keys
            $BITCOIN_PGP_KEY \
        && wget -qO bitcoin.asc $BITCOIN_ASC_URL \
        && gpg --verify bitcoin.asc \
        && tar -xzvf $BITCOIN_DIST -C /usr/local --strip-
            components=1 --exclude=*-qt \
        && rm bitcoin* \

        # remove build dependencies
        && apt-get purge -y --auto-remove wget

ENV BITCOIN_DATA /data
RUN mkdir $BITCOIN_DATA \
        && chown bitcoin:bitcoin $BITCOIN_DATA \
        && ln -s $BITCOIN_DATA /home/bitcoin/.bitcoin
VOLUME /data
COPY docker-entrypoint.sh /entrypoint.sh
ENTRYPOINT ["/entrypoint.sh"]

EXPOSE 8332 8333 18332 18333
COPY bitcoin.conf /home/bitcoin/.bitcoin/bitcoin.conf
COPY bitcoin.conf /root/.bitcoin/bitcoin.conf
```