# Utveckling av blockchain - Ett första steg in

Thomas Frödin Larsson, Anton Eliasson Gustafsson, Jonathan Uhre, Riaz Safdar

7 januari 2020

## Sammandrag

En Blockchain är ett decentraliserat nätverk där noderna i nätverket samverkar med konsensus av data som finns i så kallade block vilka är länkade med kryptografiska metoder. Tillämpningsområden för blockchain-nätverk är flera där det mest uppmärksammade är kryptovalutor. Tillämpningsområdena utöver kryptovalutor är dock inte standardiserade i varken kommersiella eller allmännyttiga tjänster. Företag som Cybercom är därför intresserade av hur en slutprodukt kan se ut, problem under utvecklingen, samt hur den kan tillämpas i Amazon Web Services (AWS). Ett verktyg för att lyckas tillämpa en blockchain i AWS är AWS-tjänsten Amazon Managed Blockchain (AMB). AMB stödjer i sin tur Hyperledger Fabric (HLF) som är en utvecklingsplattform för blockchain. Plattformen är till stor del ett paket med filer och skript. För att göra sig bekant med HLF och AMB användes informationsinsamling och genomgångar från olika hemsidor. I genomgångarna upptäcktes flera otydligheter som blev förhinder i utvecklingen. Ofta var lösningarna svåra att hitta och i vissa fall hittades de inte alls. Dessutom samverkade HLF's skriptfiler på ett sätt som lätt kunde bli överväldigande att förstå under en 8-veckors tid. I slutändan gick det att skapa en implementation av en blockchain med HLF, men i AWS uppstod problem som inte kunde lösas.

# Innehållsförteckning

# 1 Inledning

Under de senaste 10 åren har en marknad av kryptovalutor växt fram som förlitar sig på kryptografiska metoder, nämligen blockchain-teknologi [1]. Teknologin var till en början en nisch för just kryptovalutor, men har sedan dess utforskats mer och har fått uppmärksamhet i andra tillämpningsområden. Den främsta fördelen anses vara garantin om säkerhet och validering i avsaknad av en central aktör [25].

Trots popularitet inom teknikvärlden är det en nästan orörd del av mjukvaruutveckling inom IT-branschen.

## 1.1 Syfte

Företag som Cybercom har inte mycket kunskap om vad en blockchain är. De har inte heller någon kunskap i hur utvecklingen av en blockchain går till. Syftet med projektet är därför att ta ett första steg in och ge insikt i vad som påträffas under utvecklingen (exempelvis olika problem), hur en slutprodukt kan se ut, samt om en blockchain kan köras på molntjänst. Det ska tjäna som grund för ett eventuellt framtida arbetslag så att de snabbare kan påbörja utvecklingen.

## 1.2 Avgränsningar

Ett av önskemålen från projektgivaren Cybercom var att få upp en blockchain på molntjänst. Vidare kunde Cybercom bara ge tillgång till verktyg inuti molntjänsten Amazon Web Services (AWS), vilket innebär att val av moltjänst blev avgränsat till just AWS [22]. AWS har stöd för blockchain på deras molntjänst om blockchain-nätverket skrivs med utvecklingsvertyget Hyperledger Fabric (HLF)[22]. Därav avgränsas val av blockchain-plattform till HLF. På grund av den korta tiden för projektet togs ett beslut om att inte använda någon lärobok, men också eftersom djupgående kunskap om blockkedjor ej eftersöktes.

## 1.3 Frågeställningar

- Går det att visa ett körbart exempel på en blockchain med hjälp av plattformen HLF?
- Vad finns det för problem eller svårigheter med utvecklingen av en blockchain med hjälp av HLF?
- Hur körs en blockchain i molntjänsten AWS?

# 2 Metod

Projektet präglades av informationssamling och genomgångar som finns tillgängliga på olika hemsidor. Detta gällande både HLF och AWS. Informationen som insamlades delades i kommunikationskanalerna Slack och Discord. Vidare diskuterades informationen vid möten som hölls två eller tre gånger per vecka.

## 2.1 Teknisk arbetsgång

En stor del av projektet gick ut på att undersöka HLF. Efter att ha laddat ner nödvändig programvara och filer, körs kommandon i kommandoprompten i syftet att få igång ett blockchain-nätverk. Vilka kommandon som skulle köras kom från genomgångar tillgängliga på olika hemsidor. Det blev relevant att undersöka dessa kommandon och hur de intereagerade med såväl andra program som olika skriptfiler. Mycket av insikten vad gäller den tekniska aspekten av arbetet kom härifrån. Emellertid kunde det bli en överväldigande uppgift att sätta sig in på grund av många samverkande skriptfiler i olika programspråk varav de flesta hade hundratals rader kod. Att följa genomgångar med olika kommandon gällde AWS också. För implementationen i AWS var det stundom enklare att konfigurera eftersom det fanns ett gränssnitt på hemsidan för viss funktionalitet. Å andra sidan, för andra inställningar och konfigurationer krävdes körning av långa kommandon och effekter och/eller syftet av kommandot var inte var helt tydliga.

## 2.2 Felsökning

Inte sällan var flertalet HLF-genomgångar utdaterade, vilket genererade olika felaktigheter i kommandoprompten. Dessa felaktigheter var tvugna att lösas. Då kommandoprompten ger ett felmeddelande kan detta uppsökas. Felsökningen gjordes då ofta via Google där man kunde hitta mer information om felmeddelandet. Ibland hittades tydliga och konkreta lösningar, men långt ifrån varje gång. I AWS kunde det knappt hittas några lösningar via googling, förmodligen eftersom verktyget är så nytt men även att det är så ovanligt. I vissa fall blev det då nödvändigt att gissa och testa sig fram, både gällande HLF och AMB.

## 2.3 Källkritik

Mycket informationssamling har kommit ifrån HLF's samt AWS's egna hemsidor. Då mycket av informationen var mängdmässigt stor och tekniskt komplex uppstod ibland osäkerheter i hur olika tekniska aspekter faktiskt håller ihop. Med andra ord var vissa delar av den insamlade informationen otydlig.

# 3 Teknisk bakgrund

De teoritekniska delarna som är centrala i projektet är blockchain, HLF och AWB. HLF är en plattform för blockchain-utveckling och AWS är en molntjänst som har stöd för HLF.

## 3.1 Blockchain

En blockchain används för att lagra information om transaktioner. Varje transaktion lagras som ett block på en kedja i kronologisk ordning. Det ligger stort fokus på säkerhet med begrepp som kryptografi och decentralisering i centrum.

### 3.1.1 Uppbyggnad

En blockchain är i grunden en kedja av block. I varje block lagras information om en transaktion, ett datum och en hash till föregående block i kedjan. En hash är resultatet av en hashfunktion, som beräknar ett kryptografiskt värde av data. Det lagras vanligtvis som en hexadecimal sträng och blir då mindre storlek än den okrypterade datan. I en blockchain beräknas hash utifrån all information som lagras i föregående block, inklusive hashen som lagras där. Det gör att varje block påverkas av alla tidigare block i kedjan. Om ett block ändras måste alla efterliggande block anpassas för att kedjan ska vara intakt. Det är till för att göra kedjan oföränderlig. [7]

### 3.1.2 Decentraliserad lagring

En blockchain ska vara decentraliserad. Det betyder att kedjan inte ska lagras på en central server, utan istället lagras hos varje nod i nätverket. Idén är att ingen ska äga kedjan och kunna ändra i den utan andras samtycke. När en transaktion har genomförts skickas ett broadcast-meddelande ut på nätverket i form av ett block. Alla noder ska då spara det blocket som bevis för att transaktionen har ägt plats, även kallat validering. Den sanna kedjan baseras på kedjan majoriteten av användare har lagrat. [8]

### 3.1.3 Smarta kontrakt

Smarta kontrakt är ett protokoll som används av samma anledning som man använder vanliga kontrakt. Skillnaden är att smarta kontrakt skrivs i kod. De är både decentraliserade och automatiserade. För att göra en transaktion måste kontraktet signeras av alla inblandade parter. När kontraktet är signerat sker transaktionen automatiskt. Tilliten läggs då på kontraktet och säkra transaktioner kan göras utan behov av en tredje part. [9]

## 3.2 Hyperledger Fabric

The Linux Foundation påbörjade 2015 ett projektinitiativ med syftet att främja och fortskrida blockchain-teknik. Tanken var att flera olika branscher ska kunna

utnyttja teknologin i en plattformsmiljö där kollaboration uppmuntras. Ur detta skapades Hyperledger Fabric (HLF), agerandes som ett ramverk och grund åt organisationer som är intresserade av att bygga peer-to-peer nätverk med blockchain-teknik[2]. Källkoden innehåller bland annat färdiga script som hjälper utvecklare att initiera dessa typer av nätverk. Plattformen har en öppen källkod och är således öppen att tillgå för alla.

### 3.2.1 Ledger

En ledger kan liknas med en transaktionslogg, med både ett nuvarande värde efter godtycklig transaktion, samt en logg med transaktioner. I HLF byggs en ledger upp av två olika, ihopkopplade delar. Den första delen är något som kallas för world state, vilket är en databas som innehåller enkelt åtkomliga (nuvarande) värden. Ett world state kan förändras då något värde skapas, uppdateras eller tas bort. Alla förändringar hos ett world state sparas i en transaktionslogg. Transaktionsloggen är den andra delen av en ledger i HLF. Denna del är signifikant eftersom transaktionsloggen också är en blockchain, där varje transaktion - en förändring hos world state - sparas som ett nytt block[3].

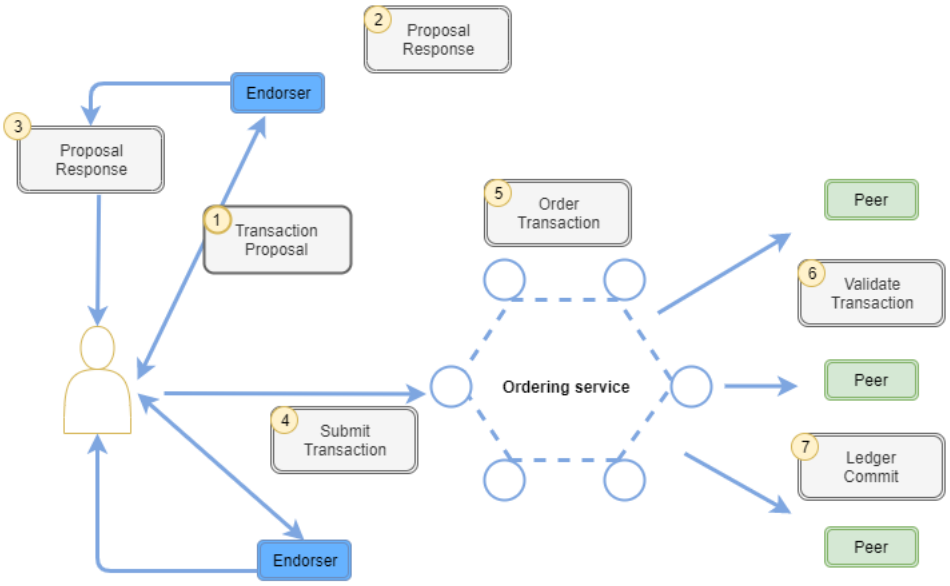
### 3.2.2 Nätverket

Likt ett konsortium kan flera organisationer på förhand bestämma vilka regler som ska gälla ett blockchain-nätverk för att sedan skapa och ansluta till det. Dessa regler kan anpassas åt organisationerna på sådant sätt att två skilda organisationer har olika behörigheter. Behörigheterna en organisation har kan emellertid ändras i efterhand, givet de andra organisationernas godkännande. Behörigheterna kan exempelvis gälla hur många noder en organisation får ha, samt vad dessa noder i sin tur kan göra. Vissa organisationer kan ges behörighet att skapa så kallade kanaler inuti nätverket. En kanal innebär en avskild ledger. Detta kan vara användbart om exempelvis två organisationer i nätverket vill dela konfidentiell information enbart med varandra. Detta innebär att en organisation kan vara ansluten till flera ledgers i ett och samma nätverk. Den här typen av uppsättning skiljer sig från andra blockchain-plattformar. Vanligtvis finns inte stöd för flera kanaler. En annan viktig skillnad mot andra plattformar och existerande nätverk är att HLF ger möjligheten att skapa privata blockchains, tillskillnad från en blockchain där vem som helst kan ansluta[4].

### 3.2.3 Transaktionsflöde i Hyperledger Fabric

När en klient försöker ändra på world-state i ett nätverk triggas smarta kontrakt hos ett subnät av noder som simulerar transaktionen mot deras egna kopior av ledgern. Detta subnät av noder kallas också för endorsing nodes. Värt att förtydliga är att ingen endorsing node ännu försöker tillämpa transaktionen mot deras respektive ledger då det enbart simuleras. Därefter får klienten ett svar (endorsed transaction proposal response) från ett set av endorsing nodes. Klienten överlämnar sedan transaktionen med svar till en så kallad ordering

service node som i sin tur tar emot flera av dessa transaktioner parallellt från andra eventuella klienter. I större nätverk är dessa typer av noder flera och jobbar kollektivt med att forma the ordering service, vars uppgift är att arrangera överlämnade transaktioner till en väldefinierad sekvens och paketera dem till block. Ett block innehåller nu en strikt sekvens av transaktioner. Blocket skickas slutligen ut till alla involverade noder på kanalen, där varje nod självständigt validerar blocket. Givet att transaktionerna i blocket har blivit godkända uppdateras ledgern med ett nytt block[5].




Figur 1. Hyperledger Fabric-high level transaction flow. Från [23]. Omarbetad med tillstånd.

### 3.2.4 Utvecklingsmiljö

Hyperledger Fabric är ett ramverk och agerar som grund åt utvecklare. Vad som ges ut av HLF är ett paket med kodfiler där ett flertal filer utgör delar av exempel på olika blockchain-nätverk. Vad gäller språket hos dessa kodfiler är de olika beroende på i vilket syfte de utgör. Tillgångarna man vill definiera i en blockchain, alltså den data som kan förändras mellan transaktioner, skrivs huvudsakligen i JSON. De smarta kontrakten (för att definiera på vilka sätt tillgångarna kan ändras) kan med hjälp av kontinuerligt uppdaterade SDK'er skrivas i Java, Node.js eller Golang. Alla körbara exempel finns som Node.js. Initiering av nätverk med organisationer, kanaler, noder, ledger, nodhantering mm. sker genom shell-script som i sin tur triggar andra script skrivna i bland annat JSON, Golang och Python.

### 3.2.5 Nödvändiga förutsättningar för utvecklaren

För att lyckas testa och köra färdiga exempelnätverk som byggs med hjälp av HLF behöver några olika programvaror vara förinstallerade. Det rekommenderas att ha kvar dessa om man som utvecklare vill bygga något på egen hand[6]. Dessa listas och förklaras kort nedan.

- Docker

  Docker kan liknas med en virtuell maskin, men det finns väsentliga skillnader. En virtuell maskin emulerar ett helt operativsystem, medan docker enbart emulerar en operativsystemkärna. Effekten blir att docker inte kräver lika mycket resurser som en virtuell maskin. Docker ger en användare möjlighet att köra programpaket i så kallade containers. Alla containers körs isolerat ifrån varandra i en och samma systemkärna. De kan emellertid kommunicera med varandra. En användare kan skapa en docker image av ett programpaket och sedan dela den till andra användare. Effekten blir då att alla användare kan köra programpaketet (givet att det körs via docker) oavsett vilket operativsystem och eventuella inställningar de har[15].

- Curl

  Curl tillhandahåller ett kommandoradsverktyg och programbibliotek för dataöverföring. Kommunikationsprotokoll som stöds är ett flertal, exempelvis FTP, HTTP med flera[16].

- Node.js & Node Package Manager (NPM)

  Node.js är ett objektorienterat programmeringsspråk som körs genom JavaScript-runtime-mijö utan en browser (till skillnad från JavaScript)[17].

  NPM är en pakethanterare för JavaScript-runtime-mijö[18].

- Go Programming Language (Golang)

  Ett objektorienterat programmeringsspråk, designat av Google[19].

- Python

  Ett objektorienterat programmeringsspråk[20].

## 3.3 Amazon Web Services

Det är samlingsnamnet för Amazons olika molntjänster. Bland andra finner man Amazon S3 som är enkel fillagring. Amazon EC2 som är cloud computing, dvs en server man har tillgång till via SSH som kör ett valfritt operativsystem. Amazon VPC(Virtual Private Cloud) som är ett eget privat nätverk (VPN).

### 3.3.1 Amazon Managed Blockchain

En av Amazons molntjänster för att administrera och hantera en blockchain. I avsaknaden av denna tjänst skulle en man behöva ha en egen server för att uppnå samma ändamål, sk. self-hosting.

# 4 Systemkonstruktion

Systemkonstruktionen kan ses som två delar, där den första behandlar konstruktionen av ett exempelnätverk med HLF som initieras på en enskild linuxmaskin. Den andra delen handlar istället om hur ett annat exempelnätverk med HLF kan köras på molntjänsten AMB.

## 4.1 Lokalt Exempelnätverk i Hyperledger Fabric

Ett färdigbyggt exempel visar hur en applikationer kan interagera med en ledger som innehållandes information om bilar. För att undvika överväldigande information om alla skriptfiler kommer enbart det smarta kontraktet och applikationskod redovisas. För den intresserade finns alla kodfiler att tillgå via fabric-samples [10].


Figur 2. Application interaction with ledger. Från [24]. Återgiven med tillstånd.

### 4.1.1 Smart kontrakt

Det smarta kontraktet som ska installeras hos noderna är skrivit i Node.js och har följande definition:

```
const Contract = require('fabric-contract-api');
class FabCar extends Contract { ... }
```

Klassen FabCar ärver klassen Contract som är en central del av HLF. Contractklassen är alltså det utvecklingsverktyg (eller SDK) som gör det möjligt att skriva de smarta kontrakten. Inuti klassen definieras en ledgers tillgångar, vilka instansieras via JSON. Dessa tillgångar består utav bilnummer, bilmärke, bilmodell, färg och ägare. Tio stycken tillgångar är hårdkodade från start. Dessutom finns det metoder inuti klassen som definerar vilka ändringar (eller transaktioner) mot ledgern en nod i nätverket får göra. Dessa metoder är:

```
async queryCar(ctx, carNumber){ ... }
async createCar(ctx, carNumber, make, model, color, owner){ ... }
async queryAllCars(ctx){ ... }
```

**async changeCarOwner(ctx, carNumber, newOwner){ ... }**

För att se det smarta kontraktet i sin helhet, se fabcar.js [11].

### 4.1.2 Applikationskod

Applikationen är i huvudsak två kodfiler skrivna i Node.js. En användare kan köra dessa genom anrop i kommandoprompten. Den första filen, query.js, låter en användare läsa av world state i ledgern. I filen finns bland annat följande rad kod:

**const result = await contract.evaluateTransaction('queryAllCars');**

Exekvering av raden ger utskrift av world state. I detta fallet - information om alla bilar. Parametern ('queryAllCars') kan bytas ut till ('queryCar',carNumer) där carNumer är ett bilnummer som finns i world state. Exekvering av detta skriver istället ut informationen om en bil. Hela skriptet finns att läsa i **query.js** [12].

Den andra filen, **invoke.js**, låter en användare skicka transaktioner mot ledgern. Notera följande kod som finns i filen:

**await contract.submitTransaction('changeCarOwner', 'CAR9', 'Dave');**

Exekvering av raden innebär att byta bilägare för CAR9 till Dave. En användare kan lätt ändra på både bil och namn inuti skriptet. Så länge bilnumret och namnet finns i world state exekveras raden. För att skapa en ny bil byts parametern ut till ('createCar', ...) där följande argument efter 'createCar' är den nya informationen om bilen som skapas. Hela skriptet finns att läsa i **invoke.js** [13].

### 4.1.3 Installation

Detta avsnittet är en genomgång i installationen av blockchain-nätverket. Alla kommandon körs i linuxmiljö.

Installera nödvändig programvara först.

- Curl
- Docker
- Go Programming Language
- Node.js & NPM
- Python

Ladda därefter ner HLF's källkod. Källkoden finns att tillgå via fabric [14].

För en detaljerad genomgång i hur man laddar ner och installerar nödvändig programvara och HLF-filer korrekt, se Bilaga 1.

Nästa steg är att orientera sig till mappen där HLF ligger. Väl där, orientera till fabric-samples/fabcar. Öppna kommandopromten mappad hit.

Nu kan man via kommandoprompten köra olika skript som finns i mappen. För att få igång nätverkets grundläggande delar; kör då via kommandoprompten:

**./startFabric.sh javascript**

Skriptet kommer att trigga andra skript; generera noder, en kanal, en organisation och chaincode (i Node.js) som installeras hos dessa. Utöver detta körs också en rad andra skript som är grundläggande delar för att få nätverket att funka i enlighet med den utformingen som HLF bygger på, exempelvis skript för ledger och nodhantering. Skripten som anropas körs i en egna docker containers.

Kör sedan kommandot:

**npm install**

Kommandot kommer trigga en JSON-fil som innehåller flera olika dependencies till vår applikation. Bland annat ger detta möjligheten att generera identiteter som kan använda denna blockchain genom transaktionsförfrågningar.

Byt sedan mapp i kommandoprompten till **fabcar/javascript** och kör följande kommandon:

**node registerUser.js**

Nu finns det en användare med en nyckel till vår ledger. Med denna användare är det möjligt att intereagera med blockchain-nätverket via två applikationsfiler, query.js och invoke.js.

## 4.2 Implementation i Amazon's moln

För implementation av en blockchain i AWS används flertalet av dess tjänster, nämligen: EC2, VPC och AMB. VPC används för att skapa ett privat nätverk vari en linuxserver från EC2 och en blockchain från AMB länkas samman. För att göra implementationen följdes Amazon's egen guide[21] för detta.
Vid val av region i Management Console för både EC2 och AMB måste North Virginia(us-east-1) väljas. Efter att ha skapat en 'instance' i EC2 behövs systemet uppdateras och flertalet program behöver installeras.

**sudo yum update -y**
**sudo yum install -y emacs telnet docker python3-pip libtool libtool-ltdl-devel git**
**sudo service docker start**
**sudo usermod -a -G docker ec2-user**
**sudo curl -L https://github.com/docker/compose/releases/download/1.20.0/docker-compose-‘uname -s‘-‘uname -m‘ -o /usr/local/bin/docker-compose**
**sudo chmod a+x /usr/local/bin/docker-compose**
**wget https://dl.google.com/go/go1.10.3.linux-amd64.tar.gz**
**tar -xzf go1.10.3.linux-amd64.tar.gz**
**sudo mv go /usr/local**

Kommandot **aws** ger tillgång till alla olika molntjänster man kan komma åt hos AWS. För att använda AMB körs kommandot **aws managedblockchain ...**. På en nystartad server är aws-kommandot en version som är över ett år gammalt och man behövder uppdatera detta.

**pip3 install --user awscli**

Det lägger sig i **/.local/bin** och måste köras från den mappen eftersom den inte skriver över programmet i **/usr/bin**.
Ytterligare behöver programspråket **go** variabler konfigurerade. Detta läggs till i **/.bash_profile**, och kör sedan **source /.bash_profile**.

**PATH=$PATH:$HOME/.local/bin:$HOME/bin**
**export GOROOT=/usr/local/go**
**export GOPATH=$HOME/go**
**export PATH=$GOROOT/bin:$PATH**

Konfigurering av aws-kommandot krävs. De olika nycklarna erhölls när ens AWS konto skapades.

**aws configure ->**
**AWS Access Key ID [None]: ***
**AWS Secret Access Key [None]: *****
**Default region name [None]: us-east-1**
**Default output format [None]: table**

Skapa en security group och applicera den på den VPC som redan existerar(default VPC). Detta för att servern ska kunna kommunicera med en blockchain. Detta görs i EC2 Management Console -> Security Groups -> Create security group. Skapa gruppen och välj Edit under Inbound. Välj All traffic under Type. Under Source välj Custom och skriv in namnet på gruppen och välj den. Spara

ändringarna. Applicera denna grupp på server genom att högerklicka på den i EC2 management console och ställ in Security Group.
Skapa blockchain-nätverk.

```
aws managedblockchain create-network \
--network-configuration '{"Name":"MyTestTaigaNetwork", \
"Description":"MyTaigaNetDescription", \
"Framework":"HYPERLEDGER_FABRIC","FrameworkVersion": "1.2", \
"FrameworkConfiguration": {"Fabric": {"Edition": "STARTER"}}, \
"VotingPolicy": {"ApprovalThresholdPolicy": {"ThresholdPercentage": 50, \
"ProposalDurationInHours": 24, \
"ThresholdComparator": "GREATER_THAN"}}}' \
--member-configuration '{"Name":"org1", \
"Description":"Org1 first member of network",\
"FrameworkConfiguration":{"Fabric":{"AdminUsername":"AdminUser","AdminPassword":"Password123
```

Detta returnerar ett network ID och member ID som kommer att användas.

För att en blockchain ska kunna koppla upp till servern måste en endpoint skapas. Detta görs i Managed Blockchain Console. Välj det nyss skapade nätverket och tryck på View Details. Välj Create VPC Endpoint och välj default VPC. Välj default subnet och välj den Security Group som skapats tidigare. Create.

Hämta programvaran för HLF.

**go get -u github.com/hyperledger/fabric-ca/cmd/...**
**cd /home/ec2-user/go/src/github.com/hyperledger/fabric-ca**
**git fetch**
**git checkout release-1.2**
**make fabric-ca-client**
**cd /home/ec2-user**
**git clone https://github.com/hyperledger/fabric-samples.git**

För att sedan starta HLF CLI (Command Line Interface) används docker-compose. En förberedlse för detta är att skapa en fil vid namn **docker-compose-cli.yaml** i **/home/ec2-user/**, vars innehåll finns som bilaga.

**docker-compose -f docker-compose-cli.yaml up -d**

En certifikatfil krävs för att kunna skapa en administatörsanvändare. Denna är universell för AMB och hämtas från Amazons servrar.

**aws s3 cp s3://us-east-1.managedblockchain/etc/managedblockchain-tls-chain.pem /home/ec2-user/managedblockchain-tls-chain.pem**

För att skapa användaren krävs en så kallad **CAEndPoint** som återfinns genom

följande kommando. network ID och member ID är från tidigare då nätverket skapades.

**aws managedblockchain get-member**
**--network-id n-XXXXXXXXXXXXXXXXXXXXXXXXXX**
**--member-id m-YYYYYYYYYYYYYYYYYYYYYYYYYY**

Admin username och password som valdes tidigare ingår även i följande kommando.

**fabric-ca-client enroll**
**-u https://AdminUsername:AdminPassword@CAEndpointAndPort**
**--tls.certfiles**
**/home/ec2-user/managedblockchain-tls-chain.pem -M**
**/home/ec2-user/admin-msp**

Mer administration av certifikat krävs.

**cp -r admin-msp/signcerts admin-msp/admincerts**
**FABRIC_CA_CLIENT_HOME=/home/ec2-user/admin-msp**

En nod i nätverket skapas.

```
aws managedblockchain create-node \
--node-configuration '{"InstanceType":"bc.t3.small","AvailabilityZone":"us-east-1a"}' \
--network-id n-XXXXXXXXXXXXXXXXXXXXXXXXX \
-member-id m-YYYYYYYYYYYYYYYYYYYYYYYYY
```

Nästa steg i guiden orsakade ett fel. Guiden instruerar att fil vid namn **configt.yaml** (se bilaga) ska skapas och ett kommando körs. Det underliga är att guiden refererar till en mapp i **/opt** vilken inte har refererats tidigare i guiden och för övrigt tillhör root, som kan vara ett problem. Flera olika kombinationer av försök gjordes genom att bland annat ändra mapp till där configtx.yaml-filen låg och att lägga till en separat variabel för att referera till mappen där filen ska ligga.

**docker exec cli configtxgen**
**-outputCreateChannelTx /opt/home/mychannel.pb**
**-profile OneOrgChannel -channelID mychannel**
**–configPath /opt/home/**

Referera till guiden[21] på steg 6 för mer information.

# 5 Resultat

Resultatet visar ett fungerande körbart exempel på en blockchain med hjälp av plattformen HLF genom att köra applikationsskript lokalt på en enskild linuxmaskin. Det uppstod dock problem under projektets gång. Både vad gäller HLF och AWS.

## 5.1 Exempelimplementation av Hyperledger Fabric

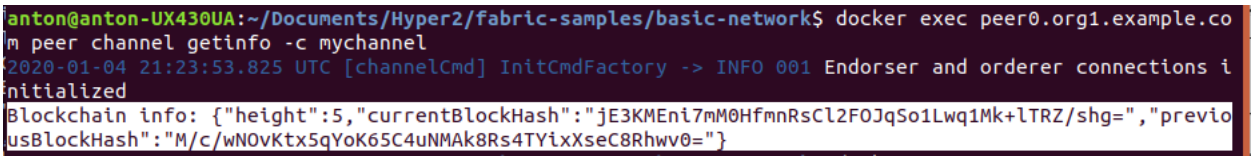
Efter installation enligt avsnitt 4.1.3 kan ett körbart exempel exekveras. Efter kommandot **node query.js** i kommandoprompten fås följande utskrift:


Figur 3. Skärmdump-1 av kommandoprompt.

Utskriften som syns i Figur 3 är world state från ledgern - alla bilar och information om dessa.

I en ytterligare kommandoprompt mappad till basic-network i HLF-mappen kan följande docker-kommando skrivas:

**docker exec peer0.org1.example.com peer channel getinfo -c mychannel**




Figur 4. Skärmdump-2 av kommandoprompt.

Figur 4 visar vad som skrivs ut efter docker-kommandot. En kort beskrivning av blockchain-status där 'height' är antalet block.

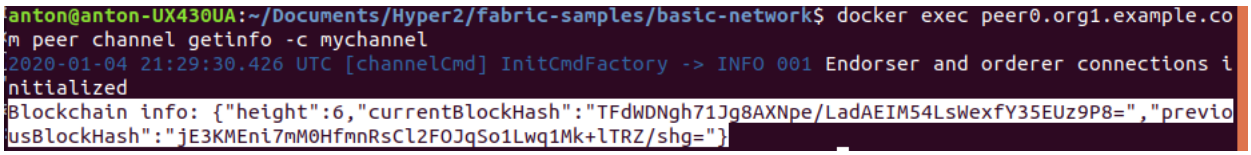
För att skicka en transaktionsförfrågan skrivs **node invoke.js** i kommandoprompten. Transaktionen försöker ändra bilägare på CAR9 från Shotaro till

Dave.

Genom att undersöka world state igen, alltså genom att exekvera **node query.js** igen, syns följande:


Figur 5. Skärmdump-3 av kommandoprompt.

Notera Figur 5. Bilen har bytt ägare.




Figur 6. Skärmdum-4 av kommandoprompt.

Efter transaktionen visar Figur 6 blockchain-status (genom att köra docker-kommandot igen) att denna blockchain har modifierats. Notera att antalet block har inkrementerats, samt att dess förgående hash är förra blockets nuvarande.

## 5.2 Problematiken med Hyperledger Fabric

Att redogöra för hur alla kodfiler som körs hänger samman hos ett exempelnätverk är en uppgift som kan bli svår. Detta eftersom mängden samverkande skriptfiler (där de flesta har hundratals rader kod) blir överväldigande att begripa under en 8-veckors period. Det smarta kontraktet var ett av de skripten som var betydligt enklare att begripa, och enklast att modifiera.

Ibland samverkar HLF med program som exempelvis Docker, vilket i sig är ett program där det kan behövas tid för att uppnå en gedigen förståelse. För att undersöka vissa skript hos HLF måste man ibland nå dessa via docker då de körs i docker containers. Exakt hur detta uppnås och egentligen fungerar är något som inte kan besvaras.

Då det uppstod buggar och felmeddelanden i kommandoprompten kunde man

ofta felsöka, men lösningen var inte alltid konkret och tydlig. Att lära sig HLF genom genomgångar som finns på olika hemsidor, kan vara lärorikt - men det uppstår lätt andra fel då dessa genomgångar inte sällan är utdaterade.

## 5.3 Amazon Managed Blockchain

Den huvudsakliga referensen för att köra en HLF-blockchain i Amazons moln är Amazons egen guide[21]. Guiden är en lång sekvens av kommandon för att konfigurera en linux-server att köra en blockchain på. Tyvärr uppstod ett fel som inte gick att lösa eller kringgå i steg 6 av 8 av guiden. Flera försök gjordes för att upprepa processen genom att skapa en helt ny instans av en linux server, men utan framgång. Detta kan ses som ett talande exempel för hur omogen teknologin ännu är. Även när den är i händerna av måttligt kapabla mjukvaru-utvecklare.

# 6 Slutsats och diskussion

De frågeställningar som utgåtts ifrån har i stor mån blivit besvarade. En enkel implementation av en HLF-blockkedja har uppnåtts. En insyn i vad användning och utveckling med blockkedjor innebär kan även demonstreras. Däremot blev implementationen av en blockkedja i Amazons molntjänst AMB ej uppnådd.

Att konfigurera och sätta upp en blockchain är en stor utmaning i nuläget. De guider som finns tillgängliga har brister i tillförlitlighet. Delvis eftersom det ofta är enskilda personer som skapar dem och att fel eller oklarheter inte korrigeras. Men även att de verktyg och skript som används för att skapa en blockchain har blivit utdaterade och på så vis inkompatibla i många fall.

## 6.1 Hyperledger Fabric

Vad som kan visas är en slutprodukt av en blockchain lokalt på egen linuxmaskin där en nod kan interegarea med en blockchain. Vad som påträffades för att uppnå ett lokalt exempelnätverk i HLF var många samverkade skript. Många stora samverkande skript skrivet på olika programspråk, försvårade insikten i hur hela plattformen funkar. Till hjälp finns förvisso mycket information att hämta, inte minst från HLF's egna hemsida - men mängden teknisk komplex information kan ta tid att absorbera. Genom att kombinera olika kommandon kunde slutproduktet genereras, men ibland fungerade inte vissa kommando då genomgångarna som beskrev dem var utdaterade. Att leta reda på olika lösningar kunde då bli tidkrävande.

Ett skript som är centralt är det samarta kontraktet, vilket definerar vilka regler som ska gälla mellan transaktionerna. För att skriva egna smarta kontrakt finns tydliga SDK'er att utgå ifrån. Andra skript, framförallt de som initierar noder, nodhantering, ledger, kanaler mm. kan däremot bli en uppgift som tar betydligt längre tid att förstå sig på, jämfört med det smarta kontraktet. Detta eftersom de samverkande skripten var skriva i flera olika språk med (ofta) hundratals rader kod. Vad som hade varit intressant att undersöka är hur man kan anpassa antalet noder, kanaler och organisationer. Dessutom hade det varit intressant att undersöka de skript som initierar detta för att konfirmera att plattformen fungerar så som det är beskrivet.

## 6.2 Hyperledger Fabric i AWS

Körning av en blockkedja i AWS gjordes ej till fullo, då den implementationen inte helt lyckades. Detta på grund av ett fel som inte kunde korrigeras. Det som däremot kan konstateras är att både nätverket som kör blockchain och verktyget för att hantera blockchain båda körs i AWS, med hjälp av EC2 respektive AMB. Vi har ställt oss från om vad poängen är att köra hela system på ett ställe(centralt) då mycket av främjandet av blockkedjor bygger på att det

är decentraliserat. Detta är dels utanför detta arbetes syfte, men skulle också kräva mycket mer kunskap eller efterforskning som vi inte besitter.

# Referenser


[1] Ledger SAS, "A brief history on Bitcoin & Cryptocurrencies", 2019. [Online]. Tillgänglig: https://www.ledger.com/academy/crypto/a-brief-history-on-bitcoin-cryptocurrencies/, hämtad: 2019-11-25.

[2] Hyperledger, "Introduction", 2019. [Online]. Tillgänglig: https://hyperledger-fabric.readthedocs.io/en/release-1.4/whatis.html, hämtad: 2019-12-10.

[3] Hyperledger, "Ledger", 2019. [Online]. Tillgänglig: https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html, hämtad: 2019-12-10.

[4] Hyperledger, "Blockchain Network", 2019. [Online]. Tillgänglig: https://hyperledger-fabric.readthedocs.io/en/release-1.4/network/network.html, hämtad: 2019-12-10.

[5] Hyperledger, "The Ordering Service", 2019. [Online]. Tillgänglig: https://hyperledger-fabric.readthedocs.io/en/release-1.4/orderer/ordering_service.html, hämtad: 2019-12-10.

[6] Hyperledger, "Prerequisites", 2019 [Online]. Tillgänglig: https://hyperledger-fabric.readthedocs.io/en/release-1.4/prereqs.html, hämtad: 2019-12-10.

[7] Horizen, "BLOCKCHAIN AS A DATA STRUCTURE", 2019. [Online]. Tillgänglig: https://academy.horizen.global/technology/advanced/blockchain-as-a-data-structure/, hämtad: 2019-12-19.

[8] A. Rosic, "Smart Contracts: The Blockchain Technology That Will Replace Lawyers", 2017. [Online]. Tillgänglig: https://blockgeeks.com/guides/what-is-blockchain-technology/, hämtad: 2019-12-19.

[9] A. Rosic, "What is Blockchain Technology? A Step-by-Step Guide For Beginners", 2017. [Online]. Tillgänglig: https://blockgeeks.com/guides/smart-contracts/, hämtad: 2019-12-19.

[10] J. Yellick et al., "fabric-samples"(Version FAB-8602), *GitHub*, 2018 [Källkod]. Tillgänglig: https://github.com/hyperledger/fabric-samples/tree/release/, hämtad: 2019-01-03.

[11] J. Zhang, "fabcar.go"(Version FAB-5260), *GitHub*, 2017 [Källkod]. Tillgänglig:

https://github.com/hyperledger/fabric-samples/blob/release/chaincode/fabcar/fabcar.go, hämtad: 2019-01-03.

[12] N. Gaski, J. Zhang, G. Singh och Herrisob, "query.js"(Version FAB-6568), *GitHub*, 2017 [Källkod]. Tillgänglig: https://github.com/hyperledger/fabric-samples/blob/release/fabcar/query.js, hämtad: 2019-01-03.

[13] G. Singh, N. Gaski, J. Zhang, Herrisob och R. Asara, "invoke.js"(Version FAB-6568), *GitHub*, 2017 [Källkod]. Tillgänglig: https://github.com/hyperledger/fabric-samples/blob/release/fabcar/invoke.js, hämtad: 2019-01-03.

[14] M. Lisrini et al., "fabric"(Version FAB-14083), *GitHub*, 2019 [Källkod]. Tillgänglig: https://github.com/hyperledger/fabric, hämtad: 2019-01-03.

[15] Docker Inc., "What is a Container?", 2020 [Online]. Tillgänglig: https://www.docker.com/resources/what-container, hämtad: 2019-01-04.

[16] D. Stenberg, "Curl", 2020 [Online]. Tillgänglig: https://www.docker.com/resources/what-container, hämtad: 2019-01-04.

[17] OpenJS Foundation, "About Node.js", 2020 [Online]. Tillgänglig: https://nodejs.org/en/about/, hämtad: 2019-01-04.

[18] npm, Inc., "What is npm?", 2020 [Online]. Tillgänglig: https://devdocs.io/npm/getting-started/what-is-npm, hämtad: 2019-01-04.

[19] The Go Authers, "Documentation", 2009 [Online]. Tillgänglig: https://golang.org/doc/, hämtad: 2019-01-04.

[20] Python Software Foundation, "Python", 2020 [Online]. Tillgänglig: https://www.python.org/about/, hämtad: 2019-01-04.

[21] Amazon Web Services Inc., "Amazon Managed Blockchain", 2019 [Online]. Tillgänglig: https://docs.aws.amazon.com/managed-blockchain/latest/managementguide/managed-blockchain- hämtad: 2020-01-07.

[22] Amazon Web Services Inc., "Amazon Managed Blockchain", 2019 [Online]. Tillgänglig: https://aws.amazon.com/managed-blockchain/, hämtad: 2020-01-07.

[23] Y. Manevich, A. Barger och Y. Tock, "Service Discovery for Hyperledger Fabric", 2018 [Online]. Tillgänglig: https://www.researchgate.net/figure/Hyperledger-Fabric-high-level-transaction-flow_fig2_3 hämtad: 2019-01-07.

[24] Hyperledger, "Writing Your First Application", 2019 [Online]. Tillgänglig: https://hyperledger-fabric.readthedocs.io/en/release-1.4/write_first_app.html, hämtad: 2020-01-07.

[25] A. Gazdecki, "How Secure Is Blockchain Technology?", *Forbes*, 12 oktober 2018 [Online]. Tillgänglig: https://www.forbes.com/sites/forbestechcouncil/2018/10/12/how-secure-is-blockchain-techno hämtad: 2020-01-07.

# Bilagor

## Bilaga 1: Hyperledger Fabric egenskriven installationsgenomgång

```
--Prerequisites, install samples and docker images--

--Efter detta är du redo för att hoppa på tutorial direkt utan att installera något.
--INSTALLERA OCH KÖR DETTA FÖRST--

sudo apt-get install curl
sudo apt-get install golang-go
export GOPATH=$HOME/go
export PATH=$PATH:$GOPATH/bin
sudo apt-get install nodejs
sudo apt-get install npm
sudo apt-get install python
sudo apt-get install docker
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb_rel
sudo apt-get update
apt-cache policy docker-ce
sudo apt-get install -y docker-ce
sudo apt-get install docker-compose
sudo apt-get upgrade

--KÖR FÖLJANDE KOMMANDO OCH SE VAD DU FÅR--

docker run hello-world

--OM DU FÅR PERMISSION DENIED--

sudo usermod -a -G docker $USER
--STARTA OM DATRON--
--PROVA docker hello-world IGEN FÖR ATT SE ATT DU INTE BLIR DENIED
```

```
--FORTSÄTT SEDAN MED DETTA--
--BYT DIRECTORY DIT DU VILL HA HYPERLEDGER FABRIC--
--EXEMPELVIS..--

cd Documents

--OCH SLUTLIGEN--
curl -sSL http://bit.ly/2ysbOFE | bash -s
```

### Bilaga 2: docker-compose-cli.yaml

```
version: '2'
services:
  cli:
    container_name: cli
    image: hyperledger/fabric-tools:1.2.0
    tty: true
    environment:
      - GOPATH=/opt/gopath
      - CORE_VM_ENDPOINT=unix:///host/var/run/docker.sock
      - CORE_LOGGING_LEVEL=info # Set logging level to debug for more verbose logging
      - CORE_PEER_ID=cli
      - CORE_CHAINCODE_KEEPALIVE=10
    working_dir: /opt/gopath/src/github.com/hyperledger/fabric/peer
    command: /bin/bash
    volumes:
        - /var/run/:/host/var/run/
        - /home/ec2-user/fabric-samples/chaincode:/opt/gopath/src/github.com/
        - /home/ec2-user:/opt/home
```

### Bilaga 3: configtx.yaml

```
################################################################################
#
#   Section: Organizations
#
#   - This section defines the different organizational identities which will
#   be referenced later in the configuration.
#
################################################################################
Organizations:
    - &Org1
            # DefaultOrg defines the organization which is used in the sampleconfig
            # of the fabric.git development environment
        Name: MemberID
```

```
            # ID to load the MSP definition as
        ID: MemberID
        MSPDir: /opt/home/admin-msp
            # AnchorPeers defines the location of peers which can be used
            # for cross org gossip communication.  Note, this value is only
            # encoded in the genesis block in the Application section context
        AnchorPeers:
            - Host:
              Port:

################################################################################
#
#   SECTION: Application
#
#   - This section defines the values to encode into a config transaction or
#   genesis block for application related parameters
#
################################################################################
Application: &ApplicationDefaults
        # Organizations is the list of orgs which are defined as participants on
        # the application side of the network
    Organizations:

################################################################################
#
#   Profile
#
#   - Different configuration profiles may be encoded here to be specified
#   as parameters to the configtxgen tool
#
################################################################################
Profiles:
    OneOrgChannel:
        Consortium: AWSSystemConsortium
        Application:
            <<: *ApplicationDefaults
            Organizations:
                - *Org1
```