upGrad

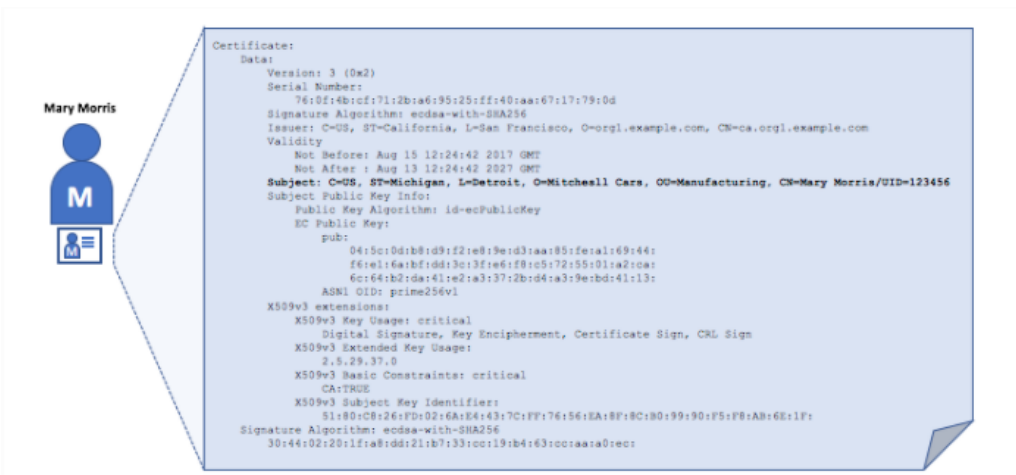
Interview Questions: Chaincode Development

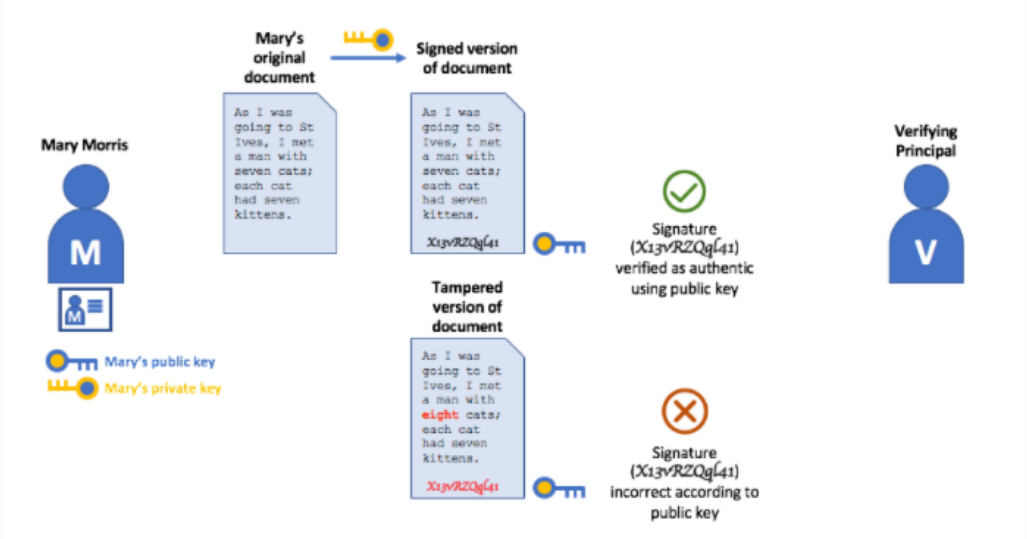
Q1	What is chaincode?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/Fabric-FAQ.html#chaincode-smart-contracts-and-digital-assets
Ans	A chaincode is a programmatic code that is deployed on the network. It is run and validated by the chain validators together during the consensus is achieved. Developers use chaincodes to develop business contracts, asset definitions, and collectively-managed decentralised applications.

Q2	How to create a business contract?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/Fabric-FAQ.html#chaincode-smart-contracts-and-digital-assets
Ans	There are two ways to build business contracts. First way is to write individual contracts into standalone instances of chaincode; the second and the more efficient way, is to use the chaincode to create decentralised applications that can manage the life cycle of one or more types of business contracts, and let the end-users instantiate instances of contracts within these applications.

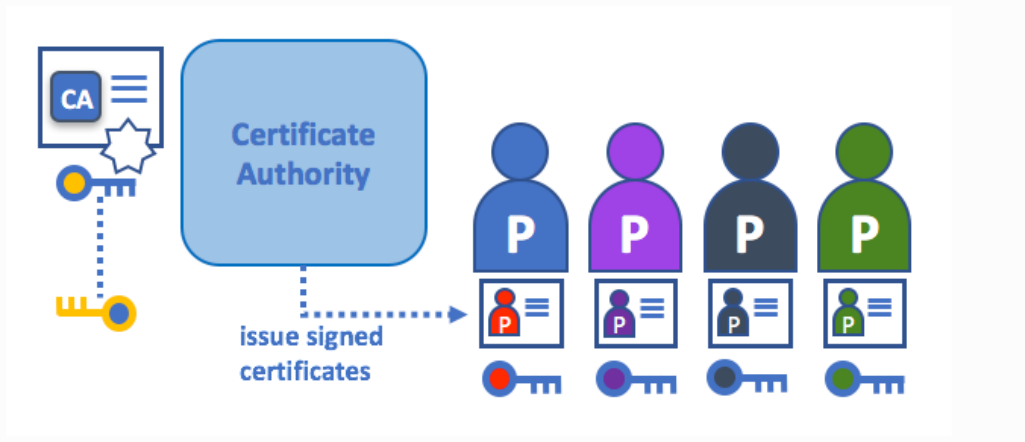
Q3	How to create an asset?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/Fabric-FAQ.html#chaincode-smart-contracts-and-digital-assets
Ans	The users can use membership service (for digital tokens) and chaincode (for business rules) to create assets, as well as the logic behind them. There are two general approaches to defining assets in most blockchain solutions: the stateless UTXO model, where the account balances are

	encoded into past transaction records; and the account model, where the account balances are stored in state storage space on the ledger. Each approach has its own benefits and problems. This blockchain technology does not prefer either one over the other. Instead, one of the first requirements is to ensure that both approaches can be implemented easily.

Q4	What is the standard of the certificate used in Hyperledger Fabric?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/identity/identity.html#digital-certificates
Ans	A digital certificate is used to verify the users of the network. It is as follows:  *A digital certificate describing a party called Mary Morris. Mary is the* `SUBJECT` *of the certificate, and the highlighted* `SUBJECT` *text shows key facts about Mary. The certificate also holds many more pieces of information, as you can see. Most importantly, Mary's public key is distributed within her certificate, whereas her private signing key is not. This signing key must be kept private.*

Q5	How to verify certificates?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/identity/identity.html#digital-certificates
Ans	The traditional authentication mechanisms rely on digital signatures that, as the name suggests, allow a party to digitally sign its messages. The digital signatures also guarantee the integrity of the signed message. 

Q6	What is a certificate authority (CA) and how does it help in permission management in Hyperledger Fabric?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/identity/identity.html#digital-certificates
Ans	CAs are responsible for distributing certificates to the nodes of the network. In some cases, digital identities (or simply identities) have the form of cryptographically validated digital certificates that comply with X.509 standard and are issued by a certificate authority (CA).

	 *A Certificate Authority dispenses certificates to different actors. These certificates are digitally signed by the CA and bind together the actor with the actor's public key (and optionally with a comprehensive list of properties). As a result, if one trusts the CA (and knows its public key), it can trust that the specific actor is bound to the public key included in the certificate, and owns the included attributes, by validating the CA's signature on the actor's certificate.*

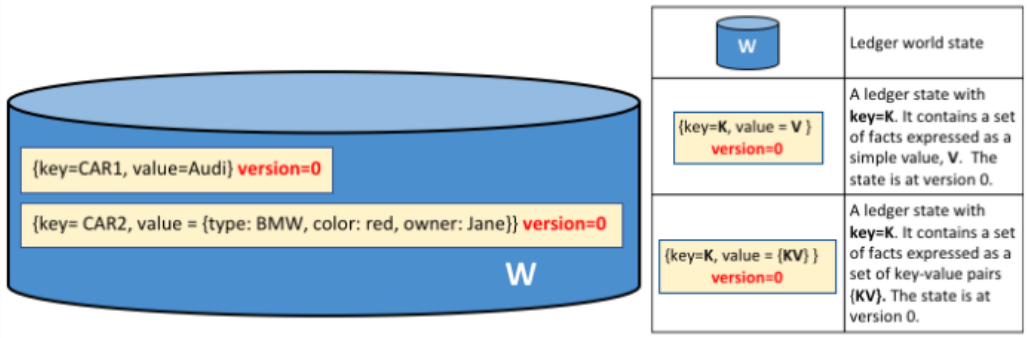
Q7	Which type of chaincode is used to define rules for the channel?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/configtx.html
Ans	A shared configuration for a Hyperledger Fabric blockchain network is stored in a collection configuration transaction, one in each channel. Each configuration transaction is usually referred to by the shorter name, 'configtx'.

Q8	What is a connection profile?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/developapps/connectionprofile.html
Ans	A connection profile describes a set of components, including peers, orderers, and certificate authorities in a Hyperledger Fabric blockchain network. It also contains the channel and organisation information relating to these components. It is primarily used by an application to configure a gateway that handles all network interactions, allowing it to

	focus on the business logic. Further, a connection profile is normally created by an administrator who understands the network topology.

Q9	Which db is used to save the world state db- CouchDB or level DB?
Ans	LevelDB is a default and is particularly appropriate when the ledger states are key-value pairs. A LevelDB ledger is closely co-located with a network node – it is embedded within the same operating system process. CouchDB is a particularly appropriate choice when the ledger states are structured as JSON documents, because CouchDB supports the rich queries and updates of richer data types that are often found in business values. Implementation-wise, CouchDB executes in a separate operating system process, still there is a 1:1 relation between a CouchDB peer node and a CouchDB instance. All of this is not visible to a smart contract.

Q10	What is a world state database?
Reference	https://hyperledger-fabric.readthedocs.io/en/release-1.4/ledger/ledger.html#world-state
Ans	A world state holds the current value of the attributes of a business object as a unique ledger state. This is useful as programs usually require the current value of an object; it would be cumbersome to traverse the entire blockchain to calculate an object's current value – you can just get it directly from the world state.



A ledger world state containing two states. The first state is: key=CAR1 and value=Audi. The second state has a more complex value: key=CAR2 and value={model:BMW, color=red, owner=Jane}. Both states are at version 0.