General Scheme of Blockchain Network in Hyperledger(how Hyperledger Fabric allows organizations to collaborate in the formation of blockchain networks):

*Concepts:

-Public/Private/Consortium Blockchains: In blockchain context there are two main questions:

1. Who controls the blockchain?
2. Who have access to read or write to the blockchain?

Here is what I've found out so far: The basic idea of introducing blockchain was to decentralize decisions and operations which is totally satisfied in “Public” chains. In public chains anyone accessing internet, can read, decide(verify) and write(be a part of consensus) to the ledger.

In the other hand, “Private” chains model traditional business into digital world. So everything is controlled(decided, operated, ruled) by a central organization. Here the central organization can decide who can read the data.

The third type of chain is “Consortium” which is partly private meaning that not anyone can decide and execute nor a central company controls everything, but a group of few approved nodes can do so whilst they must be equally involved in the consensus and other decisions.

-Permissioned vs Permissionless Blockchains: In blockchains trying to all nodes have the same ledger instance(blocks, order of blocks, order of txs in a block) is an important issue.

In permissionless blockchains any node can participate in the consensus process, wherein transactions are ordered and bundled into blocks so their consensus algo is probabilistic and this can yield forks of the ledger. Hyperledger fabric blockchain is permissioned and uses ordering nodes exclusively to make and add blocks.

-ordering nodes(ordering service): Hyperledger uses a certain set of nodes(ordering service) to order transactions and put them in a block. This action has two benefits: 1.finality of the ledger 2. performance and scalability, eliminating bottlenecks which can occur when execution and ordering are performed by the same nodes.

-Channel: A channel is a private blockchain overlay which allows for data isolation and confidentiality.

-Chaincode refers to smart contract in this context

- Invoke: channel Id, chaincode function to invoke, args[]

*A client application invokes chaincode by sending a transaction proposal to a peer. The peer will execute the chaincode and return an endorsed proposal response to the client application(Endorsement). The client application will gather enough proposal responses to satisfy an endorsement policy, and will then submit the transaction results for ordering, validation, and commit. The client application may choose not to submit the transaction results. For example if the invoke only queried the ledger, the client application typically would not submit the read-only transaction, unless there is desire to log the read on the ledger for audit purpose.

-Endorsement: **specific peers(known as Endorsing Peers)** receive proposals from a client app, execute a chaincode transaction and return response which includes: the chaincode execution response message, results<> (read set and write set), events, signature to prove the peer's chaincode execution. Chaincode applications have corresponding endorsement policies.

-Endorsing peers: The peer nodes on a channel that must ***execute transactions***(here it means 1.to write/read data on the ledger or 2.start and initialize a chaincode on a channel) attached to a specific chaincode application

-Endorsement Policy: Defines endorsing peers, and the required combination of responses (endorsements).

-Transaction: App clients gather invoke or instantiate responses from endorsing peers and package the results and endorsements into a transaction that is submitted for ordering, validation, and commit.

-*Query: A query is a chaincode invocation which reads the ledger current state but does not write to the ledger. The chaincode function may query certain keys on the ledger, or may query for a set of keys on the ledger. Since queries do not change ledger state, the client application will typically not submit these read-only transactions for ordering, validation, and commit. Although not typical, the client application can choose to submit the read-only transaction for ordering, validation, and commit, for example if the client wants auditable proof on the ledger chain that it had knowledge of specific ledger state at a certain point in time.