upGrad

#LifeKoKaroLift

Managed Services and Integrations

upGrad

Lecture on: Managed Services and Integrations

Instructor: Jaywardhan Sawale

upGrad

RECAP

- Considerations in solution deployment
- UPSTAC solution deployment

TODAY’S AGENDA

- Blockchain as a service
  - Microsoft Azure
  - Oracle managed blockchain
  - AWS managed blockchain

- Hyperledger caliper

- Session recap

BLOCKCHAIN AS A SERVICE (BaaS)

- It is a packaged solution mainly targeted at enterprise users
- Provides cloud-based solution to deploy and manage the distributed blockchain platform
- Tools to manage distributed network
- Tools to manage adding and removing organisations to the network
- Secure user identity management
- Built in services to rapidly develop and deploy solution

REQUIREMENT FOR BaaS

upGrad

- Industry tilt towards cloud based solutions
- Complexities in deployment and managing distributed solutions
- Operational overheads in deploying and maintaining blockchain infrastructure
- Higher operational costs

PRE-REQUISITES

- Choice of support for blockchain technology
- Consensus mechanisms provided
- Security and high availability
- Backend services
- Ease of use
- Stability of BaaS platform

BaaS SERVICE PROVIDERS

- AWS
  - Hyperledger fabric
- IBM blockchain platform
  - Hyperledger fabric
- Microsoft Azure
  - Ethereum
- Oracle managed blockchain
  - Hyperledger fabric
- Kaleido
  - Ethereum
  - Corda

Amazon Managed blockchain

upGrad

- Support for Hyperledger fabric and Ethereum
- Fully managed. Create blockchain network across multiple AWS accounts seamlessly.
- Easy scalability of hardware and network components
- Provides more reliability. Its ordering service QLDB maintains immutable transaction change logs.

Azure Managed blockchain

- Support for Ethereum
- Enables creation of consortia network seamlessly
- Easy integration of blockchain network to various data storage and streaming services
- Provides blockchain development kit for easy smart contract development and deployment.

Oracle Managed blockchain

upGrad

- Support for Hyperledger Fabric
- Fully managed
- Easy access to oracle cloud services

Integrating Blockchain Solutions with Other Applications

INTEGRATING BLOCKCHAIN APPLICATIONS

Integration of external applications with blockchain is still an emerging area. External application can play two types of roles, and they are as follows:

- Oracles
    - Similar to public blockchains, external applications act as a source of truth. Smart contracts, via interfaces such as Oraclize, directly talk with these application to take a decision.

- Users
    - Applications can invoke smart contracts or act on transactional event from smart contracts. In such cases, the external application needs to be provided with an identity like an individual user and it acts as one.

EVENT-BASED INTEGRATION

- For every transaction, blockchain emits events.
- These events can be captured by a module and pushed to database.
- This data can be used for general purposes in application.
- You need to access blockchain data directly if current source of truth is required.
- These events also act as trigger points for interfacing with external application.

INTEGRATING BLOCKCHAINS

Similar to external applications, which can interface with each other, blockchains can also talk to each other. This can be accomplished using a bridge.

- Sidechains on Ethereum
  - A sidechain is a blockchain network that runs applications and frequently stores its current state on public blockchain.
  - A smart contract is deployed on public blockchain that stores the current state (in most cases, number of tokens held by participants) and its mirror is deployed on sidechain.
  - A gateway interfaces between the two chains and updates between them are synced frequently.

Hyperledger Caliper Demonstration

HYPERLEDGER CALIPER

- A performance measurement tool that helps user to benchmark solutions with different use cases

- Supported platforms
  - Hyperledger fabric
  - Ethereum
  - Hyperledger Burrow
  - Hyperledger Besu
  - Hyperledger Sawtooth
  - Hyperledger Iroha

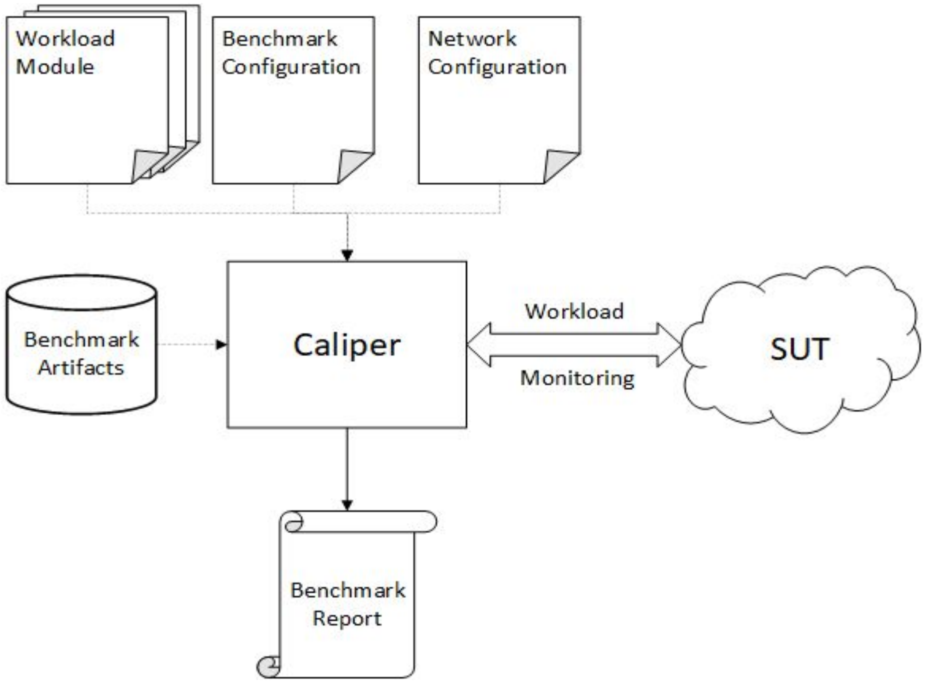
SUPPORTED METRIC

- Success rate
    - Measures total success and failures of all the transactions
- Transaction and read throughput
    - Rate of transactions in the system (tps) in a given cycle
- Transaction and read latency
    - Time taken from transaction issuance to receipt of response
    - Min, max, avg latency in a given cycle
- Resource consumption
    - Maximum and minimum resource consumption (memory, CPU, Network IO traffic) in a given cycle

Caliper Architecture

upGrad

- Caliper is a service that creates transactions against system under test and monitors its response



https://hyperledger.github.io/caliper/v0.3.2/architecture/

Benchmark Configuration

- It defines how the benchmark load is to be executed
- Has 3 sections Test, Observer and Monitor
- Test section has information about number of rounds, rate at which transactions are to be submitted
- Observer section determines how the information from worker process is gathered
- Monitor attribute determines how the monitoring of metrics captured will happen

Network Configuration

upGrad

- This file content is specific to the system under observation
- Contains information about network topology, users
- Attributes might differ based on network types

Workload Module

- Core of Caliper benchmarking tool. It is responsible for constructing and submitting transactions
- It is loaded through factory functions and should export a single factory function *createWorkloadModule()*
- Interface retired by this factory function should contain three asynchronous functions

Workload Module

upGrad

- *initializeWorkloadModule()* → Function is called by worker process before submitting each transaction. It provides transaction context

- *submitTransaction()* → This function is called by worker process each time rate controllers enables next transaction

- *cleanupWorkModule()* → It is called at the end of the round and used for resource cleanup that is implemented.

Recap

RECAP

- Identified the requirement of blockchain
- Identified user stories
- Finalised participant organisations and users
- Hyperledger fabric network design
- Identity management and wallets
- Smart contract development
- API development and transaction handlers
- Deployment to AWS nodes.
- Blockchain as a service
- Hyperledger caliper

FUNDAMENTALS OF SOLUTION DESIGN

- Is there a need for blockchain for a given solution?
- Identifying blockchain technology
  - Does solution require a public, permissioned, or private blockchain?
  - What kind of privacy considerations are we looking at?
  - What is the throughput required?
- Identity management
  - Using wallets
  - Authentication mechanisms
- Finalising on tech stack to be used
- Smart contract development
- Performance measurement

FINAL REMARKS

In this session, you learnt to deploy the solution in AWS cloud and make it accessible to general users.

Over the period of this module, you looked at the various aspect of blockchain based solution design and development. Let us summarize it in a few pointers:

- You began with a basic question of whether blockchain is the right technology to solve a particular problem.
- You then designed the solution by choosing the right blockchain technology and went ahead to implement smart contracts and API server.
- You also understood how to deploy solution in local system, followed by deployment in AWS cloud.

FINAL REMARKS

- Finally, you analysed the various considerations in solution deployment. Now that we have a working solution deployed, let’s start using it.
- Remember that the overall design and solution framework might remain the same for a solution based on one flavour of blockchain, but there are always nuances that one must be careful about.
- Suppose, in some of the deployments a third-party CA is to be used. The user cards might need to be stored in different secured vaults or peer nodes can sit on different cloud infrastructures. You need to be aware of these differences and build solution to satisfy those requirements.

upGrad

Questions???

upGrad

#RahoAmbitious

Thank You!