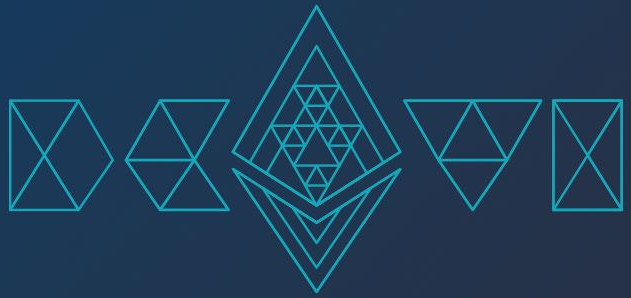


Blockchain Indexers: 101

Benjamin Memisevic

Smart Contract and Software Engineer

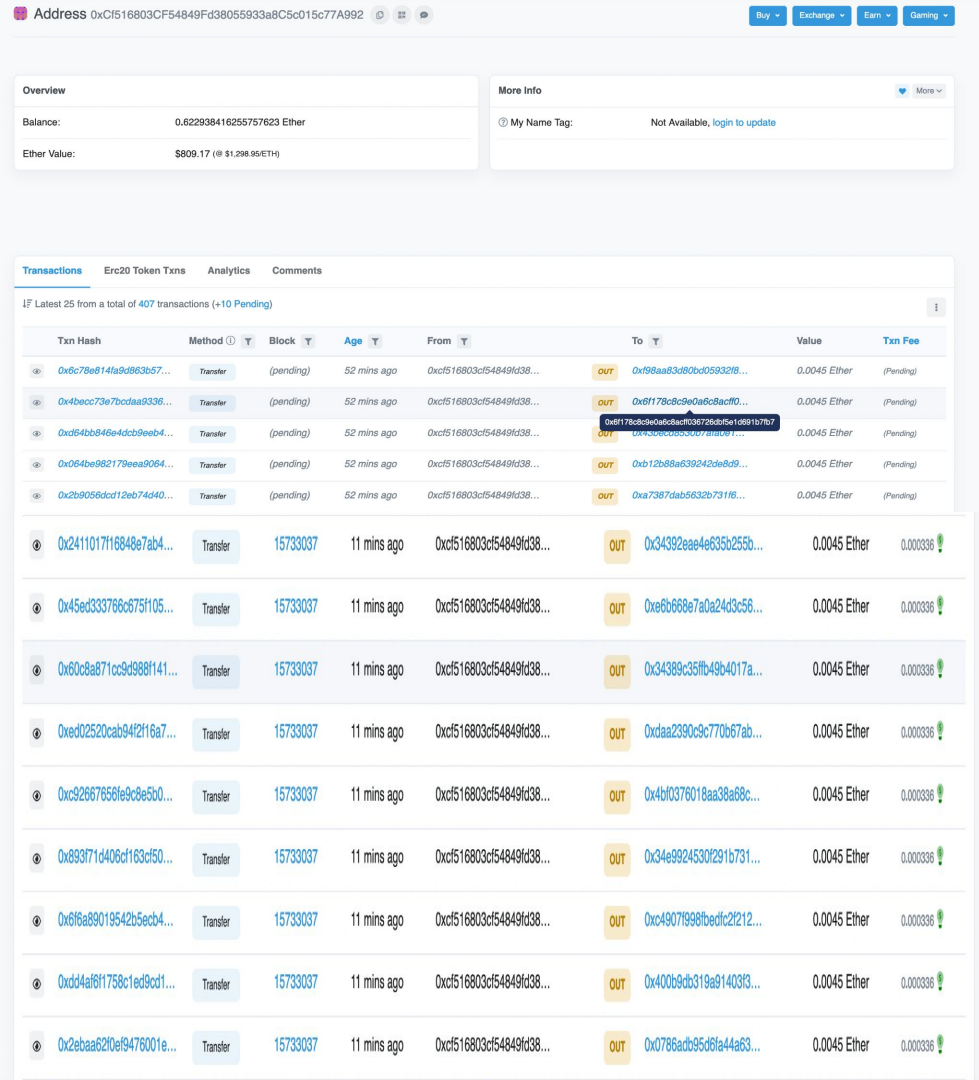
Section 1

Background

This Workshop and Me

- Worked in the space since 2018
- SC Engineer at Labrys and TracerDAO/Mycelium
- Team Lead Engineer at Mycelium delivering on-chain monitoring of oracle systems
- Spent 2 years working on Indexing and event processing pipeline solutions

Please ask questions as we go along!



Etherscan: Lots to see

- Clicking through and exploring is hugely beneficial, you'll always discover something new/interesting
- Bot attempting to quickly fanout eth to multiple addresses (not a contract but an EOA!)

Question

Observability and Transparency: Pros and Cons

Solana DeFi Trading Platform Mango Markets Loses $100M in Hack

The latest hack comes less than a week after BNB Chain lost $100 million.

Section 2

What is an Indexer?

Dune Analytics

Etherscan

the graph

Extract:Transform:Load

- Extract: Request data from a Node
- Transform: Change its form to be easily human/machine readable
- Load: Insert it to have a better retrieval strategy or pass it to another service

Why Make an Indexer?

Retrieval Time

- Creating a unique index allows quick retrieval
- Reduce node requests

Example:
Index to fetch the transaction history of an address

Non-Permanent Data

- Store mempool history for developing arbitrage strategy
- Store and simulate transactions and their results
- Gas estimation history of local development environment

Breaking Down an Example: Chainlink

- Even decoded it's difficult to tell what's going on
- Some points are obvious: answer
- Some are very confusing (i.e. observers)

Transaction Receipt Event Logs

113

Address 0xb2f68c82479928669b0487d1daed6ef47b63411e

Name: NewTransmission (index_topic_1 uint32 aggregatorRoundId, int192 answer, address transmitter, int192[] observations, bytes observers, bytes32 rawReportContext) View Source

Topics
0 0xf6a97944f31ea060dfde0566e4167c1a1082551e64b60ecb14d599a9d023d451
1 Dec → 2305

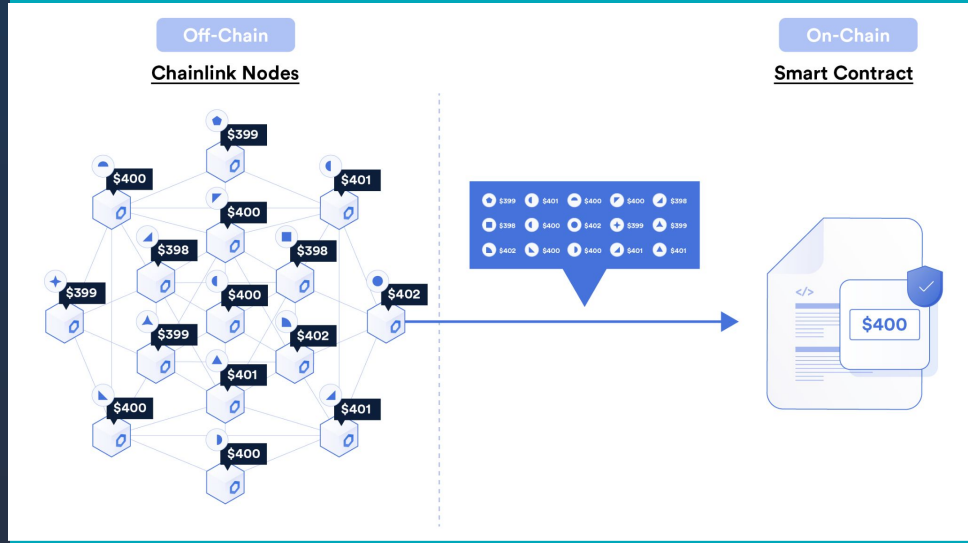
Data (Dec | Hex)

answer : 439530000000000
transmitter : 0xFa0E4F48a369BB3eCBCEe0B5119379EA8D1bcF29
observations : 439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439530000000000
439640000000000
439640000000000
439640000000000
439640000000000
439640000000000
439890319647420
441130570000000
observers : 07010A03040806000E09020B0C0D050F
rawReportContext : 0000000000000000000000AF7E7276CA516315E45364AC43E128A10003DD7004

Step 1: Understand the System and the Contracts

Chainlink: Off-Chain-Reporting

- Instead of having each node individually send a transaction with their “answer” one “**transmitter**” will send everyone's data on their behalf.
- This reduces gas usage; Chainlink used to utilize a majority of Mainnet’s bandwidth by itself!
- The transmitter also aggregates (takes a median) of everyone's answer and that is the price of the feed for that “round”



Here's the timeline.

Price Deviation

Chainlink oracles detect a price deviation and a round is initiated. All oracles assigned to the feed retrieve a price from their subscribed APIs.

Leader Collects

The round leader collects all of the different answers from each oracle

Submit On-Chain

The answer is submitted on-chain by the transmitter as well as every oracles individual answers

More Legwork:

- Observers tell us who is who
- Observations are too large to make sense: a multiplier is being used.

Transaction Receipt Event Logs

113

Address 0xb2f68c82479928669b0487d1daed6ef47b63411e

Name NewTransmission (index_topic_1 uint32 aggregatorRoundId, int192 answer, address transmitter, int192[] observations, bytes observers, bytes32 rawReportContext) View Source

Topics

0 0xf6a97944f31ea060dfde0566e4167c1a1082551e64b60ecb14d599a9d023d451

1 Dec → 2305

Data Dec Hex

answer : 439530000000000

transmitter : 0xFa0E4F48a369BB3eCBCEe0B5119379EA8D1bcF29

observations : 439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439530000000000
439640000000000
439640000000000
439640000000000
439640000000000
439640000000000
439640000000000
439890319647420
441130570000000

observers : 07010A03040806000E09020B0C0D050F

rawReportContext : 0000000000000000000000AF7E7276CA516315E45364AC43E128A10003DD7004

Adding Complexity: Block by Block Changes

24. transmitters

The list will match the order used to specify the transmitter during setConfig

0x8b1d49a93A84B5dA0917a1ed42D8a3E191C28524,0x0312EA121df0a323fF535B753172736cc9d53d13,0x218B5a7861dBf368D09A84E0dBfF6C6DDbf99DB8,0x9cFAb1513FFA293E7023159B3C7A4C984B6a3480,0xC4b732Fd121F2f3783A9Ac2a6C62fD535FD13FdA,0xFa0E4F48a369BB3eCBCEe0B5119379EA8D1bcF29,0xcC29be4Ca92D4Ecc43C8451fBA94C200B83991f6,0xc74cE67BfC623c803D48AFc74a09A6FF6b599003,0xf16e77a989529AA4C58318acEe8A1548Df3fcCc1,0x6878fb222FfF9A2fE3C0Cde77D281916f8D296b3,0xF07131F578a5F708AE2CCB9faF98458099E0FFB4,0xCe859E48f6cE9834a119Ba04FdC53c1D4F1082A7,0xe3E0596AC55Ae6044b757baB27426F7dC9e018d4,0xD22c87Dc7a3F12dcBB75CEbDA2e96f6766AE114F,0xBbf078A8849D74623e36E6DBBdC8e0a35E657C26,0xE3cd128883f2954D78923487B67Ea7C4F25C7C46 `address[]`

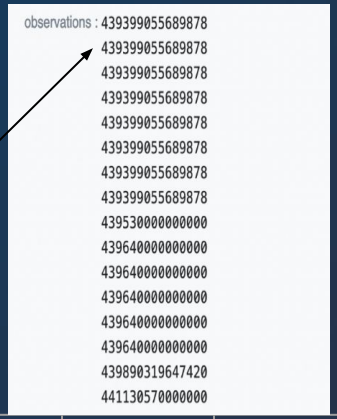
Return:

list of addresses permitted to transmit reports to this contract

4. decimals

18 `uint8`

Multipliers and transmitters can change every block!



07	01	0A	03	…	…	…	…	…	…
7	1	10	3	…	…	…	…	…	…

24. transmitters

The list will match the order used to specify the transmitter during setConfig

0x8b1d49a93A84B5dA0917a1ed42D8a3E191C28524,0x0312EA121df0a323fF535B753172736cc9d53d13,0x218B5a7861dBf368D09A84E0dBfF6C6DDbf99DB8,0x9cFAb1513FFA293E7023159B3C7A4C984B6a3480,0xC4b732Fd121F2f3783A9Ac2a6C62fD535FD13FdA,0xFa0E4F48a369BB3eCBCEe0B5119379EA8D1bcF29,0xcC29be4Ca92D4Ecc43C8451fBA94C200B83991f6,0xc74cE67BfC623c803D48AFc74a09A6FF6b599003,0xf16e77a989529AA4C58318acEe8A1548Df3fcCc1,0x6878fb222FfF9A2fE3C0Cde77D281916f8D296b3,0xF07131F578a5F708AE2CCB9faF98458099E0FFB4,0xCe859E48f6cE9834a119Ba04FdC53c1D4F1082A7,0xe3E0596AC55Ae6044b757baB27426F7dC9e018d4,0xD22c87Dc7a3F12dcBB75CEbDA2e96f6766AE114F,0xBbf078A8849D74623e36E6DBBdC8e0a35E657C26,0xE3cd128883f2954D78923487B67Ea7C4F25C7C46 `address[]`

Return:

list of addresses permitted to transmit reports to this contract

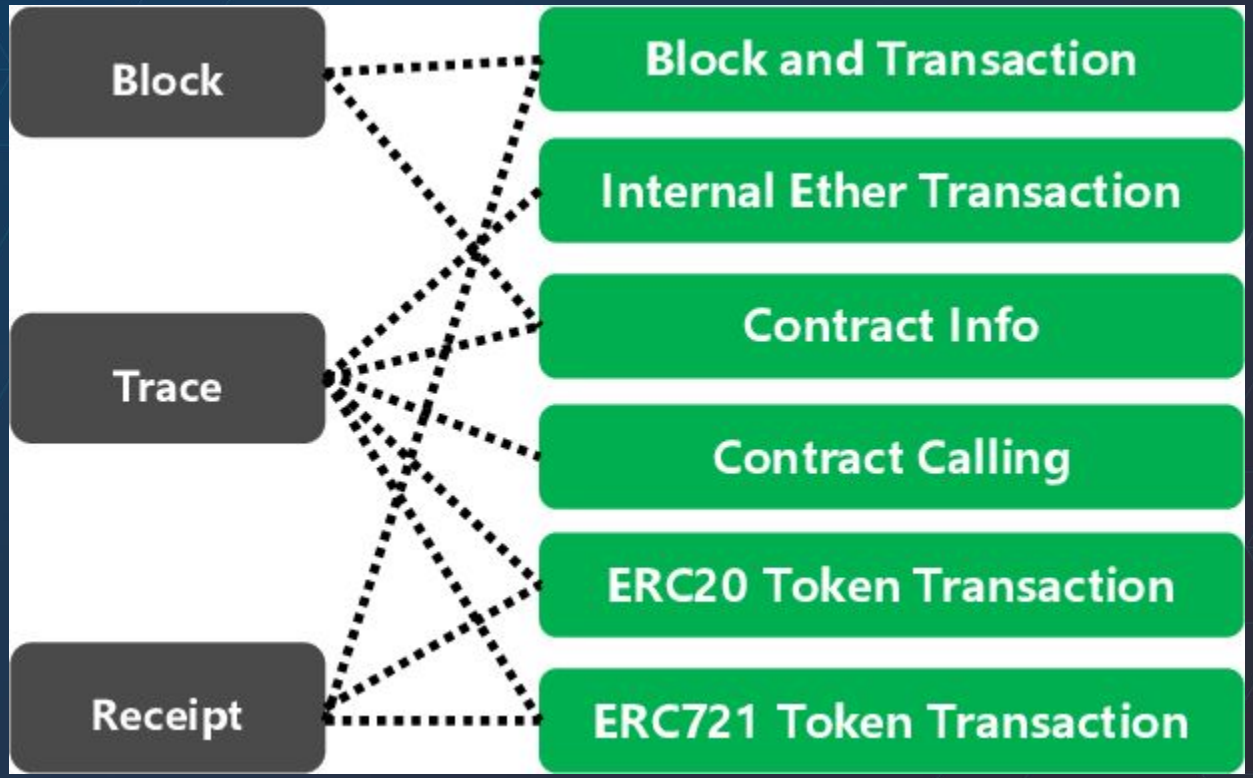
Database vs Node Retrieval

- Getting this data from a node is very complex and expensive
- Instead of all of this processing, eth calls, and fetching we could instead request this data through SQL
- Imagine trying to get the average oracle accuracy for a specific oracle on Wednesdays through node requests!

Section 2

ETH Data Types

Blocks, Transaction, Receipts, Addresses



Dissecting A Log

- Topics 1-3, Indexed data (searchable by the node)
- Data: Unlimited space
- Block Number
- Tx Index
- Log Index
- Removed

Transaction Receipt Event Logs

113

Address 0xb2f68c82479928669b0487d1daed6ef47b63411e

Name NewTransmission (index_topic_1 uint32 aggregatorRoundId, int192 answer, address transmitter, int192[] observations, bytes observers, bytes32 rawReportContext) View Source

Topics
0 0xf6a97944f31ea060dfde0566e4167c1a1082551e64b60ecb14d599a9d023d451
1 Dec → 2305

Data

Dec Hex

answer : 439530000000000
transmitter : 0xFa0E4F48a369BB3eCBCEe0B5119379EA8D1bcF29
observations : 439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439399055689878
439530000000000
439640000000000
439640000000000
439640000000000
439640000000000
439640000000000
439640000000000
439890319647420
441130570000000
observers : 07010A03040806000E09020B0C0D050F
rawReportContext : 0000000000000000000000AF7E7276CA516315E45364AC43E128A10003DD7004

Topic[0]

- This defines the log

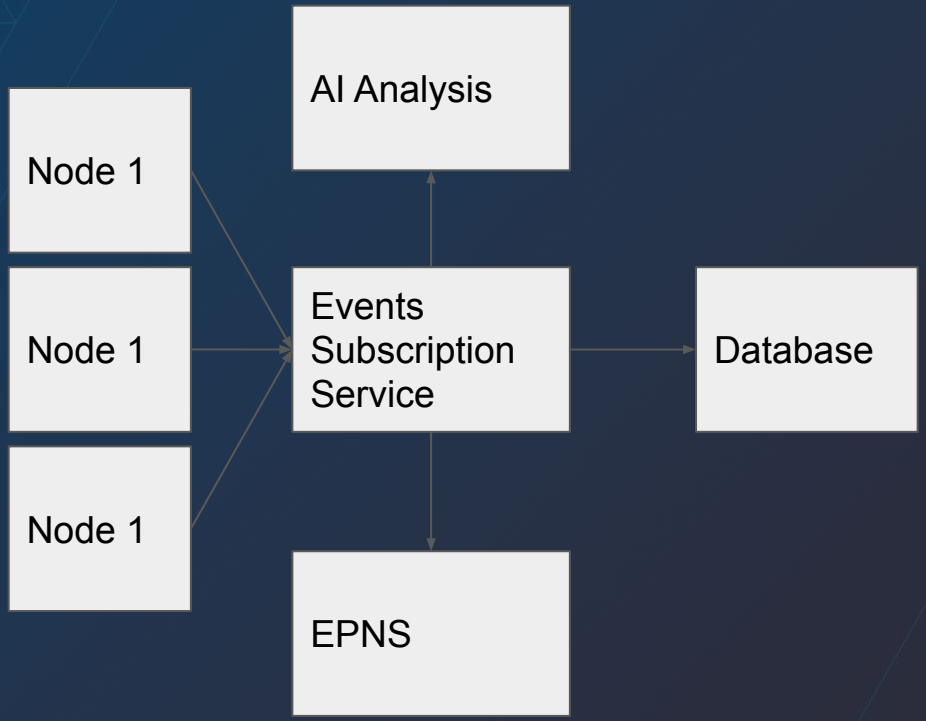
Topic[0] = kekack256(event_name(type1, type 2, ...))

Careful: This is only unique for each contract!

Storage: Private Variables are not as Private as you Think

- Can access any storage of a contract even if the variable is marked private
- Extremely useful for finding contracts using the EIP-1967 Proxy pattern

Infrastructure Design



Database Options

Timescale

Section 3

Code Walkthrough

Golang and Geth

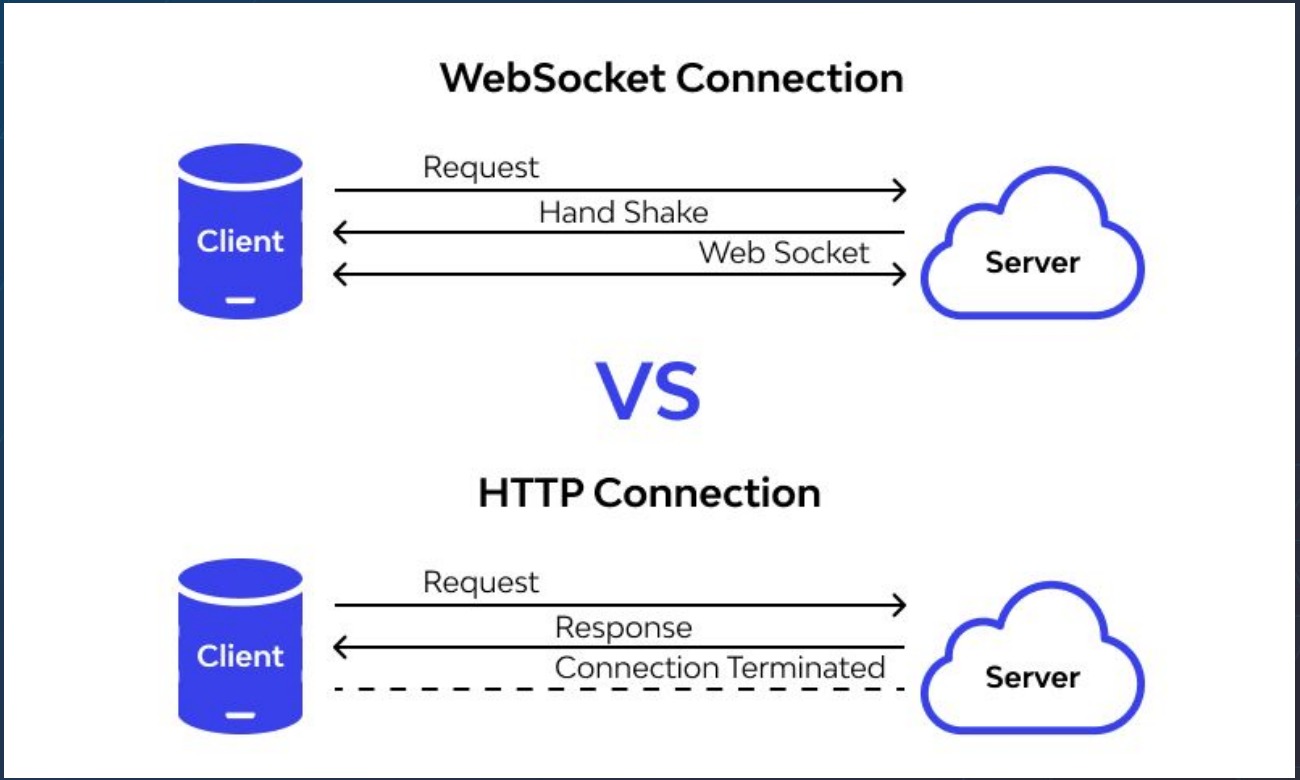
- Fast
- Extremely well maintained
- Safe parallelisation
- Geth calls are portable to most EVM chains

This will allow us to deploy our program across multiple chains since they will adhere to the RPC specification in the yellowpaper

Creating a Client

```
client, err := ethclient.Dial(rpc)
    if err != nil {
        log.Fatal("Failed to connect to the websocket of
the Node (RPC) ", err)
    } else {
        fmt.Println("successfully connected to the RPC
endpoint!")
    }
```

Websockets vs Http



Creating a Query

```
    contractAddress := common.HexToAddress("0x60Ae865ee4C725cd04353b5AAb364553f56ceF82")
    query := ethereum.FilterQuery{
        Addresses: []common.Address{contractAddress},
        Topics:    [][]common.Hash{{common.HexToHash("0x44403e38baed5e40df7f64ff8708b076c75a0dfda8380e75df5c36f11a476743")}},
    }
```

FilterLogs vs Subscribe Filter Logs

```
sub, err := client.SubscribeFilterLogs(context.Background(), query, logs)
    if err != nil {
        log.Fatal(err)
    } else {
        fmt.Println("successfully subscribed to the contract events!")
    }
```

```
historiclogs, err := clientH.FilterLogs(context.Background(), historicQuery)
```

Creating a Channel

```
logs1 := make(chan types.Log)


for {
        select {
        case err := <-sub.Err():
            log.Fatal(err)
        case vLog := <-logs1:
     //Do Processing
}
}
```

Generating an ABI

```
solc --abi events.sol

======= events.sol:Events =======
Contract JSON ABI
[{"anonymous":false,"inputs":[{"indexed":false,"internalType":"string","name":"name","type"
:"string"},{"indexed":false,"internalType":"string","name":"symbol","type":"string"},{"inde
xed":false,"internalType":"uint256","name":"timestamp","type":"uint256"}],"name":"BaseIniti
alized","type":"event"},{"anonymous":false,"inputs":[{"indexed":true,"internalType":....]}
```

Making an ABI Object and Unpacking

```
contractABI, err := abi.JSON(strings.NewReader(ABI_String))
    if err != nil {
        log.Fatal("could not convert JSON ABI string to ABI object")
    }

Interfaces, err := contractABI.Unpack("MY_EVENT_NAME", my_data)

MyBigInt := Interfaces[0].(*big.Int)
```

Working with a DB: Making Tables

```
type User struct {
    ID      string `gorm:"primaryKey"`
    Address string
}

type CommentMessage struct {
    MessageID                uuid.UUID`gorm:"primaryKey"`
    Sent                     bool
    ProfileId                decimal.Decimal
    PubId                    decimal.Decimal
    ContentURI               string
    ProfileIdPointed         decimal.Decimal
    PubIdPointed             decimal.Decimal
    CollectModule            string
    CollectModuleReturnData  string
    ReferenceModule          string
    ReferenceModuleReturnData string
    Timestamp                decimal.Decimal
}
```

Working with a DB: Connecting and Migrating

```
db, err := gorm.Open(postgres.Open(dsn), &gorm.Config{})
    //panic if we cannot connect to the database
    if err != nil {
        panic("failed to connect database")
    } else {
        //or else we are good to go
        fmt.Println("Connected to database")
        fmt.Println(db)
    }
    db.AutoMigrate(&database.User{})
    db.AutoMigrate(&database.CommentMessage{})
```

Working with a DB: Inserting and Upserting

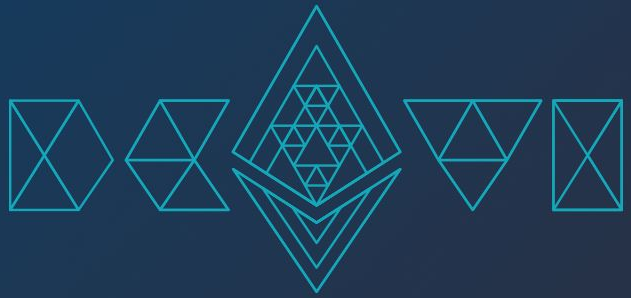
```
myvar := database.FollowMessage{
                MessageID: uuid. New(),
                Sent:       false,
                ProfileId: profileID,
                FollowNFT: followNFT,
                Timestamp: TimestampDecimal,
            }

            db. Clauses(clause.OnConflict{
                UpdateAll:  true,
            }). Create(&myvar)
```

```
ProfileIDBI := ProfileIdInterface[0].(*big.Int)
            profileID := decimal.NewFromBigInt(ProfileIDBI, 0)
            followNFT := common.HexToAddress((Topics[2].Hex())).Hex()
```

Section 4

Q&A



Thank you!

Benjamin Memisevic

@MemiHack_eth