Challenge 1. AAVE : complex and innovative ecosystem that comes with numerous risks

Introduction : A complex system with many specific concepts embedded in data

According to its documentation : Aave is a decentralised non-custodial liquidity protocol where users can participate as depositors or borrowers. Depositors provide liquidity to the market to earn a passive income, while borrowers are able to borrow in an over-collateralised (perpetually) or under-collateralised (one-block liquidity, ie Flash loans) fashion.

There is a lot of concepts that need to be understood before taking on the descriptions, monitoring and supervising the flows.

AAVE as a decentralized protocol is managed by a Decentralized Autonomous organization that takes decisions based on proposals. By protocol, one should understand multiple smart contract on a blockchain that work together to allow automated actions

We selected a few of these concepts :

- **Lending Pool**: Handles core protocol functionality (supply, borrow, withdraw, repay, flashloan, liquidationCall)
- **Oracle**: Registry of price feeds for each Aave reserve asset (Chainlink oracles)
- **Debt tokens** (S/V.Tokens): Debt tokens are interest-accruing tokens that are minted and burned on borrow and repay representing the debt owed by the token holder. There are 2 types of debt tokens: Stable debt tokens (stoken) , representing a debt to the protocol with a stable interest rate / Variable debt tokens (vtoken), representing a debt to the protocol with a variable interest rate.
- **Yield-generating tokens** : atokens are minted and burned upon deposit and withdraw. The aTokens' value is pegged to the value of the corresponding deposited asset at a 1:1 ratio, and can be safely stored, transferred or traded. All interest collected by the aTokens reserves are distributed to aTokens holders directly by continuously increasing their wallet balance.
- **Reserves** : The reserve refers to the fund of assets that are held in a smart contract, which are used to cover any potential losses or liquidation events that may occur on the platform. This reserve fund is designed to protect the protocol and the users from the impact of any unexpected market volatility or default. It is fed by deposits
- **Reserve factor** : is a percentage of interest which goes to a collector contract that is controlled by Aave governance to promote ecosystem growth

- **Liquidations** : When a loan position has a health factor below 1 meaning that it's no more overcollateralized , it can be liquidated by anyone.
- **Superuser rights** : some adresses have `RiskAdmin` or `PoolAdmin` roles

For each area of focus we will try to define the main concept before presenting the code.

1. Main indicators to follow adoption

As central banks we neeed to understand past and ongoing dynamics in adoption to asses the financial and potentieal systemic risk associated with this kind of services. To do so we focused on Total Locked value and User base

1.1 Total Value locked in USD : a useful but not perfect measure

To monitor the dynamic of the protocol we usually refer to the Total Value Locked This indicator provides information on the tokens that are locked by the users to provide liquidity to the borrowers.

As AAVE V2 is multichain (EVM compatible chains only), to be more precises it allows user to deposit token from Layer 1 (Ethereum Mainnet) and Layer 2s (Polygon, Avalanche). The total locked value already shows great difference between L1 and L2 with L2 TVL being driven by incentives (the spikes). Incentives are mostly coming from VC that distribute capital attract user (for instance Avalanche Rush : $180M liquidity mining program cleary visible as the source of the spike in AAVE AVAX)

On our presentation we count the deposited values but do not account for the token that are deposited twice or more. For instance a user can deposited ETH, borrow Wrapped ETH, swap Wrapped ETH on an exchange, then deposit again. There is also a caveat regarding the TVL in USD as it takes into account exogenous price variation that have an impact on ETH and other token price (if we consider for instance that the price variation of BTC is exogenous). It could be useful to measure the TVL in kind (in ETH, MATIC or AVAX for instance)

More work is needed to implement this double count caveat and look into TVL in kind.

To extract this data

- we used aave_deposits, aave_eth_deposits, aave_matic_deposits
- extracted the smart contract adress of all the token with aave_atokens, aave_eth_atokens, aave_matic_atokens

In [16]:
```
import itertools

import awswrangler as wr
import plotly.express as px
```

```
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

In [17]:
```
WORKGROUP = "group1_workgroup"
wr.config.workgroup = WORKGROUP
DATABASE = "hackathon_group1"
wr.config.database = DATABASE
```

In [18]:
```
import itertools

sides = ["deposits", "borrows"]
bchains = ["aave", "aave_eth", "aave_matic"]


combos = list(itertools.product(sides, bchains))


def get_amts_1(side, bchain, data_type="amounts"):
    return f"""{data_type}_{bchain}_{side} as (
    select
        t1.txhash,
        (CAST(t1.amount AS decimal) / POWER(10, CAST(t2.underlyingassetde
    from {bchain}_{side} t1
    JOIN {bchain}_atokens t2 ON SUBSTRING(t1.reserve, 1, Length(reserve)
    )"""


def get_amts_2(side, bchain, data_type="amounts"):
    return f"""select
        '{bchain}' as bchain,
        '{side}' as side,
        '{data_type}' as data_type,
        cast(from_unixtime(cast(timestamp as bigint)) as date) as obs_dat
        t2.total_amt as obs_value
    from {bchain}_{side} t1
    left join {data_type}_{bchain}_{side} t2
        on t1.txhash = t2.txhash
    """


def get_distinct_users_1(side, bchain, data_type="distinct_users"):
    return f"""{data_type}_{bchain}_{side} as (
        select
            '{bchain}' as bchain,
            '{side}' as side,
            '{data_type}' as data_type,
            cast(from_unixtime(cast(timestamp as bigint)) as date) as obs
            count(distinct user) as obs_value
        FROM
            {bchain}_{side}
        group by 1, 2, 3, 4
        order by 1, 2, 3, 4
        )
        """


def get_distinct_users_2(side, bchain, data_type="distinct_users"):
    return f"""select * from {data_type}_{bchain}_{side}"""
```

```
def get_new_users_1(side, bchain, data_type="new_users"):
    return f"""{data_type}_{bchain}_{side} as (
        select
            '{bchain}' as bchain,
            '{side}' as side,
            '{data_type}' as data_type,
            cast(from_unixtime(cast(timestamp as bigint)) as date) as obs
            count(*) as obs_value
        from (select t.*, row_number() over (partition by user order by t
        where seqnum = 1
        group by 1, 2, 3, 4
        order by 1, 2, 3, 4
        )
        """


def get_new_users_2(side, bchain, data_type="new_users"):
    return f"""select * from {data_type}_{bchain}_{side}"""


sql = f"""WITH
    {", ".join([get_amts_1(side=combo[0], bchain=combo[1]) for combo in c
    ,
    {", ".join([get_distinct_users_1(side=combo[0], bchain=combo[1]) for
    ,
    {", ".join([get_new_users_1(side=combo[0], bchain=combo[1]) for combo
    ,
    result as (
    {" union all ".join([get_amts_2(side=combo[0], bchain=combo[1]) for c
    union all
    {" union all ".join([get_distinct_users_2(side=combo[0], bchain=combo
    union all
    {" union all ".join([get_new_users_2(side=combo[0], bchain=combo[1])
    )
    select
        bchain,
        side,
        data_type,
        obs_date,
        sum(obs_value) as obs_value
    from result
    group by bchain, side, data_type, obs_date
    order by bchain, side, data_type, obs_date
    """

# print(sql)
```

In [19]:
```
%%time
df = wr.athena.read_sql_query(
    sql=sql,
    ctas_approach=False
)
```

```
CPU times: user 1.21 s, sys: 16.3 ms, total: 1.23 s
Wall time: 7.91 s
```

In [20]:
```
max_row = 3
max_cols = 1
```

```
fig = make_subplots(
    rows=max_row, cols=max_cols,
    subplot_titles=("Avalanche blockchain", "Ethereum blockchain", "MATIC
    )


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave' and side == 'deposits' and data_type ==
    y=df.query("bchain == 'aave' and side == 'deposits' and data_type ==
    name="Aave deposits amounts",
    ),
    row=1, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave' and side == 'borrows' and data_type == '
    y=df.query("bchain == 'aave' and side == 'borrows' and data_type == '
    name="Aave borrowing amounts",
    ),
    row=1, col=1)


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_eth' and side == 'deposits' and data_type
    y=df.query("bchain == 'aave_eth' and side == 'deposits' and data_type
    name="ETH deposits amounts",
    ),
    row=2, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_eth' and side == 'borrows' and data_type
    y=df.query("bchain == 'aave_eth' and side == 'borrows' and data_type
    name="ETH borrowing amounts",
    ),
    row=2, col=1)


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_matic' and side == 'deposits' and data_ty
    y=df.query("bchain == 'aave_matic' and side == 'deposits' and data_ty
    name="MATIC deposits amounts",
    ),
    row=3, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_matic' and side == 'borrows' and data_typ
    y=df.query("bchain == 'aave_matic' and side == 'borrows' and data_typ
    name="MATIC borrowing amounts",
    ),
    row=3, col=1)

fig.update_layout(height=800, width=1600,
                  title_text="Deposit and borrowing amounts by blockchain

fig.update_yaxes(type="log", row=1, col=1)
fig.update_yaxes(type="log", row=2, col=1)
fig.update_yaxes(type="log", row=3, col=1)

fig.show()
```

In [21]:

```
fig = make_subplots(
    rows=max_row, cols=max_cols,
    subplot_titles=("Avalanche blockchain", "Ethereum blockchain", "MATIC
    )


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave' and side == 'deposits' and data_type ==
    y=df.query("bchain == 'aave' and side == 'deposits' and data_type ==
    name="Aave deposits distinct users",
    ),
```

```
    row=1, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave' and side == 'borrows' and data_type == '
    y=df.query("bchain == 'aave' and side == 'borrows' and data_type == '
    name="Aave borrowing distinct users",
    ),
    row=1, col=1)


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_eth' and side == 'deposits' and data_type
    y=df.query("bchain == 'aave_eth' and side == 'deposits' and data_type
    name="ETH deposits distinct users",
    ),
    row=2, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_eth' and side == 'borrows' and data_type
    y=df.query("bchain == 'aave_eth' and side == 'borrows' and data_type
    name="ETH borrowing distinct users",
    ),
    row=2, col=1)


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_matic' and side == 'deposits' and data_ty
    y=df.query("bchain == 'aave_matic' and side == 'deposits' and data_ty
    name="MATIC deposits distinct users",
    ),
    row=3, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_matic' and side == 'borrows' and data_typ
    y=df.query("bchain == 'aave_matic' and side == 'borrows' and data_typ
    name="MATIC borrowing distinct users",
    ),
    row=3, col=1)

fig.update_layout(height=800, width=1600,
                  title_text="Deposit and borrowing distinct users by blo

fig.update_yaxes(type="log", row=1, col=1)
fig.update_yaxes(type="log", row=2, col=1)
fig.update_yaxes(type="log", row=3, col=1)

fig.show()
```

In [22]:

```
fig = make_subplots(
    rows=max_row, cols=max_cols,
    subplot_titles=("Avalanche blockchain", "Ethereum blockchain", "MATIC
    )


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave' and side == 'deposits' and data_type ==
    y=df.query("bchain == 'aave' and side == 'deposits' and data_type ==
    name="Aave deposits new users",
    ),
```

```
    row=1, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave' and side == 'borrows' and data_type == '
    y=df.query("bchain == 'aave' and side == 'borrows' and data_type == '
    name="Aave borrowing distinct users",
    ),
    row=1, col=1)


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_eth' and side == 'deposits' and data_type
    y=df.query("bchain == 'aave_eth' and side == 'deposits' and data_type
    name="ETH deposits new users",
    ),
    row=2, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_eth' and side == 'borrows' and data_type
    y=df.query("bchain == 'aave_eth' and side == 'borrows' and data_type
    name="ETH borrowing new users",
    ),
    row=2, col=1)


fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_matic' and side == 'deposits' and data_ty
    y=df.query("bchain == 'aave_matic' and side == 'deposits' and data_ty
    name="MATIC deposits new users",
    ),
    row=3, col=1)

fig.add_trace(go.Scatter(
    x=df.query("bchain == 'aave_matic' and side == 'borrows' and data_typ
    y=df.query("bchain == 'aave_matic' and side == 'borrows' and data_typ
    name="MATIC borrowing new users",
    ),
    row=3, col=1)

fig.update_layout(height=800, width=1600,
                  title_text="Deposit and borrowing new users by blockcha
fig.update_yaxes(type="log", row=1, col=1)
fig.update_yaxes(type="log", row=2, col=1)
fig.update_yaxes(type="log", row=3, col=1)

fig.show()
```

1.2. User base - A picture of the adoption of the services

There are several type of users :

- Lenders or Depositors : they acquire yield by providing liquidity to the pool (by holding atokens)
- Borrowers : they pay interest for borrowing token (a part of the interest pays the depositors and part pays the protocol)

- DApps : decentralized app may build upon AAVE to prodive their on services. For instance mai.finance allows user to use atokens to gain more yields.
- Governance token holder : they take part in the governance decisions

Measure the user base is a way to measure the ongoing adoption of the services allowing the regulator to compare it with traditionnal finance. As of now the number of user is ...

To go a little further :

Real time information

There is real time information available both for the user on its userinterface and in the datasets with Borrowing Position, Collateralization Ratio, Loan History, Health Factor, Market Information.

Aave Governance and Guardian :

AAVE is not perfectly decentralized. Some user have super user roles for emergency situations :

- The Aave Community Guardian is a group of community-elected individuals or entities who are part of a 6/10 multisig - Their role is notably to Protect the Aave Protocol against potential governance takeovers by having the ability to “veto” a proposal if it is deemed malicious.
- They can also pause Aave markets in case of emergency

KYC regarding this super user could be implemented as their identity is knwon for the most part (https://docs.aave.com/governance/aave-guardians):

- Ernesto Boado (BGD Labs)
- Matthew Graham (Governance House)
- Fernando Martinelli (Balancer Labs)
- Coderdan (Aavegotchi)
- Corbin Page (ConsenSys Codefi, Aave Grants DAO)
- Gavi Galloway (Standard Crypto)
- Meltem Demirors (Coinshares)
- Marc Zeller (Aave Companies)
- Hilmar Maximilian Orth (Gelato)
- 0xmaki

In [23]:
```
select aave_deposits.user,
SUM(CAST((CAST(aave_deposits.amount AS decimal) / POWER(10, CAST(aave_ato
COUNT(*) AS numOfDepositsPerUser
from aave_deposits
JOIN aave_atokens ON SUBSTRING(aave_deposits.reserve, 1, Length(reserve)-
GROUP BY user
```

```
  Cell In[23], line 1
    select aave_deposits.user,
           ^
SyntaxError: invalid syntax
```

2. Innovations embeded in the protocol and associated risks

Based on the AAVE documentation, the AAVE protocol recognize three kind of risks. (1) Risk related to the token that are accepted or not : smart contract security and counter-parties in the governance of the token. (2) Market risks and (3) Volatility risks.

- Smart Contract Risk : smart contract risk focuses on the technical security of a token based on its underlying code. Basically is the code includes errors.
- Counter-party Risk : different degrees of governance decentralisation that may give direct control over funds (as backing, for example) or attack vectors to the governance architecture.
- Market Risk : related to the market size and fluctuations in offer and demand of the collateral. If the value of the collateral decreases, it might reach the liquidation threshold and start getting liquidated. The markets then need to hold sufficient volume for these liquidations - sells which tend to lower the price of the underlying asset through slippage affecting the value recovered.
- The volatility risk : crypto assets can be subject to sudden volatility spikes; it is not uncommon to witness 30% changes in price within a week or a month. When this is a price increase, to protect our users, it might be followed by a parameter readjustment to limit risks of new operations.

AAVE DAOs frequently update AAVE protocol risk policies, especially when market conditions change.

The main variable for risk management in the procotols are :

- Wether the token can be used as collateral for borrowing. For instance USDT, BUSD, PAX and sUSD are considered by the protocol as strongly exposed to the risk of single point of failure in their governance. They are not admitted as collateral. This can be seen in the data :
- What is its loan to value ratio :LTV ratio defines the maximum amount of token that can be borrowed with a specific collateral
- What are he liquidation treshold and bonus : liquidation threshold is the percentage at which a position is defined as undercollateralised. For example, a Liquidation threshold of 80% means that if the value rises above 80% of the collateral, the position is undercollateralised and could be liquidated.

2.1. Credit Delegation : an innovation that does not seems to be widely adopted

Credit Delegation allows a depositor to deposit funds in the protocol to earn interest, and delegate borrowing power (i.e. their credit) to other users. One of the first example was made by DeversiFi or inside the AAVE protocol for getting WETH from ETH. The enforcement of the loan and its terms are agreed upon between the depositor and borrowers, which can be either off-chain via legal agreements or on-chain via smart contracts. This enables:

- The depositor (delegator) to earn extra yield on top of the yield they already earn from the protocol,
- The borrowers (delegatees) to access an uncollateralized loan.

It doesnt not seems that this functionnality has been used much as the total number of transaction is only 78 384 (on the 3 chains). Yet this functionnality has been brought to AAVE V3.

Data used :

- aave_eth_stabletokendelegatedallowances
- aave_eth_variabletokendelegatedallowances
- aave_matic_stabletokendelegatedallowances
- aave_matic_variabletokendelegatedallowances
- aave_variabletokendelegatedallowances

Code used :

In [ ]:
```
SELECT count(*) as transaction_no from (
    SELECT id FROM aave_eth_stabletokendelegatedallowances
    UNION SELECT id FROM aave_eth_variabletokendelegatedallowances
    UNION SELECT id FROM aave_matic_stabletokendelegatedallowances
    UNION SELECT id FROM aave_matic_variabletokendelegatedallowances
    UNION SELECT id FROM aave_variabletokendelegatedallowances)
```

2.2. Flashloans : a widely adopted innovation that came with a lot market disturbance and hacks

Flash Loans are special uncollateralised loans that allow the borrowing of an asset, as long as the borrowed amount (and a fee) is returned before the end of the block transaction. To do a Flash Loan, you will need to build a contract that requests a Flash Loan. The contract will then need to execute the instructed steps and pay back the loan + interest and fees all within the same transaction. The flashloan is not free, it cost the initiator 0.09% of the borrowed amount (which can be find in the "aave_BLOCKCHAIN_flashloans.totalfees" datasets)

Flash loans are features allowed by blockchain architecture and they have many usecases for instance arbitrage (user can quickly rebalance liquidity pools by borrowing large amount of tokens to buy and sell assets), colateral swap (a user replace its collateral in one token by another token in one transaction) or selfliquidation (to recoup liquidation fees on the collateral).

Unfortunately flash loans are also tools for hacks (not necessarly the source of it) and act as multiplier of impact as they may give virtually unlimited leverage to the attack (for instance in the example found in the database, the attacker used x68 the initial amount of token he owned). In the Inverse Finance example we took, the hack is made possible by price oracle manipulation (due to the high amount of token borrowed).

We use the available data in the dataset to look for transactions involved in hacks, which enable the supervisor to track the money (at least, up to the mixer Tornado Cash)

Data could used :

- aave_eth_flashloans.id
- aave_matic_flashloans.id
- aave_flashloans.id

Process : When the hack is already known :

- research for hack transaction hash on available databases (Defilama, Twitter, Rekt.news). For the Inverse Finance hack : https://twitter.com/peckshield/status/1537382891230883841?s=20&t=uxBt2flcD_f82MM2VD3pCg
- find the transaction hask in the id column : 0x958236266991bc3fe3b77feaacea120f172c0708ad01c7a715b255f218f9313c : https://etherscan.io/tx/0x958236266991bc3fe3b77feaacea120f172c0708ad01c7a71
- Follow the following transactions : https://etherscan.io/address/0x7b792e49f640676b3706d666075e903b3a4deec6

2.3 Liquidations : an internal mechanism mostly used by bots

According to AAVE documentation (https://docs.aave.com/faq/liquidations) a liquidation is a process that occurs when a borrower's health factor goes below 1 due to their collateral value not properly covering their loan/debt value. This might happen when the collateral decreases in value or the borrowed debt increases in value against each other. This collateral vs loan value ratio is shown in the health factor.

In a liquidation, up to 50% of a borrower's debt is repaid and that value + liquidation fee is taken from the collateral available, so after a liquidation that amount liquidated from your debt is repaid.

The database aave_eth_liquidationcalls only shows 312 different liquidators adress on Ethereum. The liquidation process is highly competitive and mostly operated by bots. This mechanism is vulnerable to market manipulation

2.4. KYC in DEFI : is it possible ?

YES ! The main avenue to implement KYC on DEFI is to follow the wallet. Most (except for Miners and airdrop receivers) user need ON/OFF ramps to acces DEFI. ONramps to convert fiat to crypto and offramps for the opposit operation. The datasets aave_eth/matic_repays.user provide information on the wallet address. These wallet can be traced back to transactions to centralized exchange (OFFramps). If the CEX are subject to proper regulation the regulator can identify the user this way.

Example from the dataset aave_eth_repays (on Ethereum):

- The user (0x6532a7c47b033e83eac702b2baecdb8b834558de) that initiated the repayment under the transaction hash 0x7d9f7a7943a56e3fa37952f13526d488c6a26b90680d21f228eaa2c6af42fc3c
- Also received a incoming Transaction from binance for 11.3 ETH 50 days ago with the transaction hash 0x8f9d8e893c7b5c0714947d73e30fa3bb86d5a292c47b5e389f37de95eaf98591
- We can even assume that this user is coming from europe (or Swiss) as he interact with DEFI Franc a Defi platform around the Franc Suisse (transaction hash 0x33deb99430a8b406ac8288062542a2c00fded400226eb778f94d36c2bd490b6

2.5. KYC embedded in the protocol ? A system that still needs to prove its efficiency

Permissionned part of AAVE : To foster adoption by institution AAVE developed a "permissioned" version of the Protocol.

In the data prodivided we were not able to find transactions on any of the contracts deployed for AAVE ARC (list here https://aave-arc.gitbook.io/docs/deployed-contracts/arc). Espacially we looked in the dataset

Aave Arc is a separate and independent deployment of the Aave Protocol smart contracts, with an addition of a smart contract permission layer which allows whitelisters to manage all participants in the pools (after KYC checks).

Conclusion for Challenge 1

AAVE is a complex ecosystem with many specific concepts embedded in the data.

The data produced by this economic activity is overwhelming. We needed not only to process the data but also to understand the underlying mechanisms and concepts. In our presentation, we focused on defining these concepts, relying in particular on the AAVE documentation. We will not go into each of them but we have selected 3 key words that came up during our two days and nights of discussions and that we believe have the biggest implications for central banks and financial regulators.

Adoption This is the conjunction between the total locked value and the number of users. Without going back over the figures, the idea here is to see that even for this

protocol which is among the most used, the order of magnitude is low compared to traditional finance. But the flow of new users is relatively constant and important. It is sometimes boosted by marketing operations to distribute tokens. Conclusion for a central bank: it is not systemic but a constant monitoring is necessary. The code we propose would allow to do this monitoring.

Innovation Among the innovations proposed by the protocols, some have not worked, such as the delegation of credit, it seems that according to the data, there is no real use. And others have worked a lot (some would say too much), such as Flashloans. They have an internal use but are also frequently used to increase the leverage of hacks.

Regulation In conclusion: what are the avenues for regulation? there is two dimensions to take into account, similarly to macro and microprudential regulation. If the monitoring of on chain data (that we presented here) and main indicators is essential from a regulatory perspective it seems necessary to go beyond that. For instance we should monitor DAOs, follow wallet and understand individual flows to properly supervise DEFI. To be precise we could follow proposals and updates to understand what is beeing prepared for the future. For instance in AAVE V2 one can do that by following the Proposal Contract and understant who is voting for what : https://etherscan.io/address/0xec568fffba86c094cf06b22134b23074dfe2252c

Also it's necessary to investigate on testnets for instance AAVE V3 is preparing a new stablecoin GHO backed by the deposits to the pool (with new concepts such as Facilitators, that remains to be investigated)

Much work is still needed on that space and we can even imagine on chain regulation embedded in smart contracts that could bridge the two dimensions of macro and micro prudential / supervision.