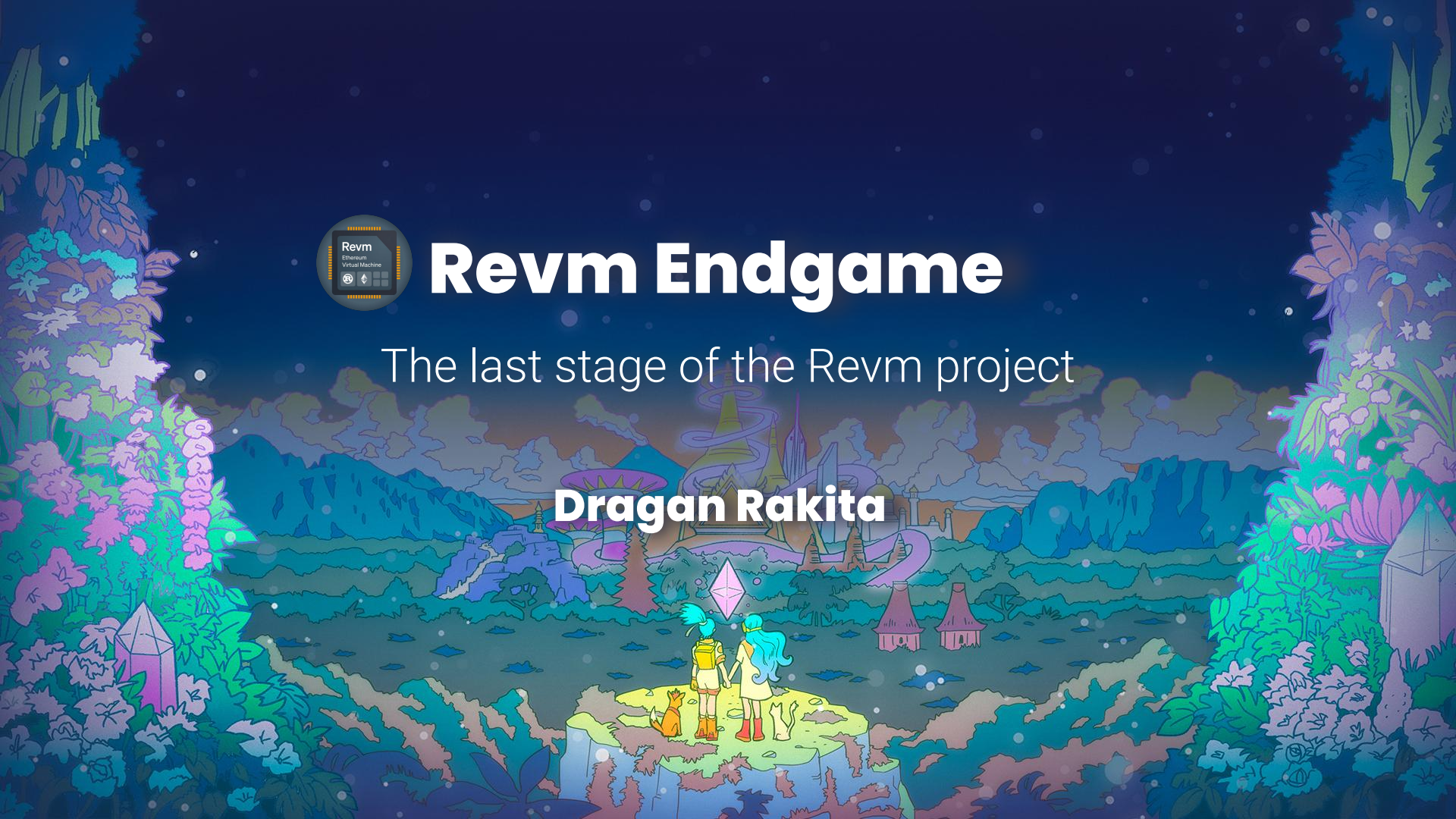


Revm Endgame

The last stage of the Revm project

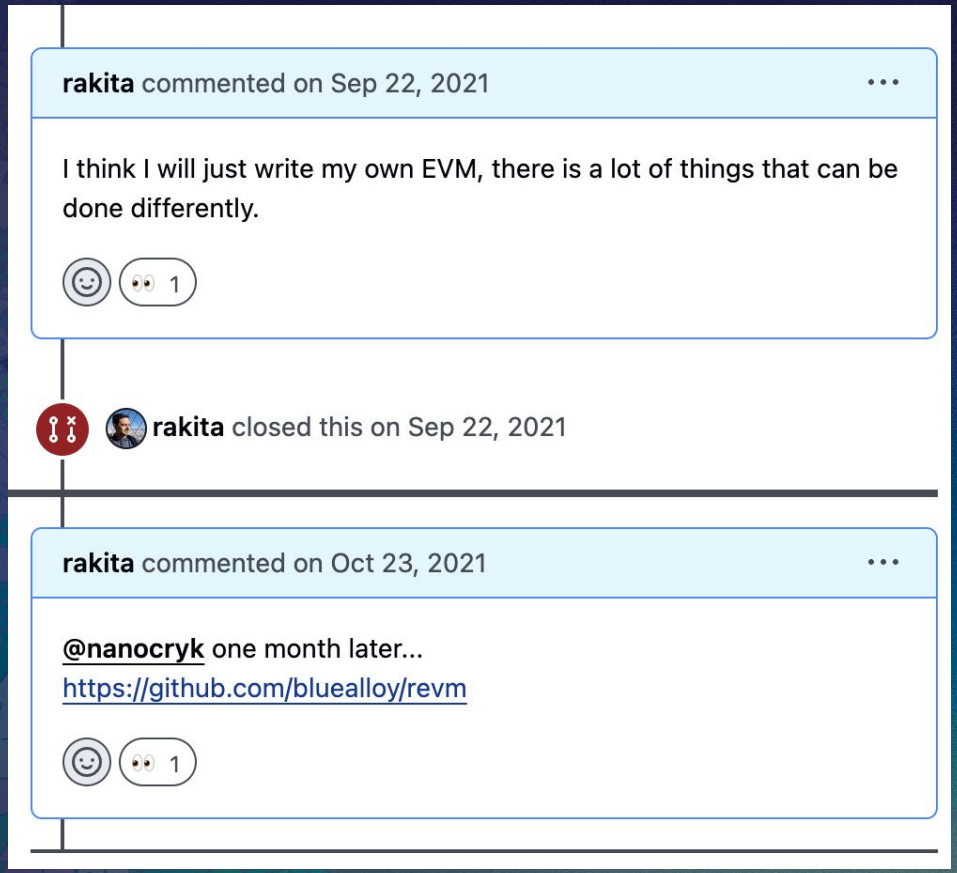
Dragan Rakita

History

History and Evolution of the project

EVM library state in 2021

- At that time there were three implementations
  - OpenEthereum had GPL licence so it was not suitable
  - evmodin was new and had exotic/experimental rust features, where I wanted to use stable rust
  - SputnikVM was not that maintained, tracing was hard to use, while journaling was not optimal and interface was strange.
- I was on a break from work (Have burned out on OpenEthereum) and few months into my break, after cooling my head, I had few ideas for a product that uses EVM and was in need of a good library.



So this happened

- At first it supported only a few latest hardforks, and even passing those tests were hard and time consuming. I had few bugs that were found out later but in general i was satisfied.
- Fun fact, foundry and Revm had started in same month, without even knowing each other

Sputnik PR: https://github.com/rust-ethereum/evm/pull/55

2021- End of the year

- **Latest forks** were added and EF statetest were passing
- It supported all **precompiles** out of box
- Interface was simple.
  - You had **Database** trait (To get runtime data) and **Env** (to set Tx/Block static data)
  - **Inspector** trait was used for runtime inspection for both the calls and instructions. This **abstraction** paid of in Foundry and to this day is very powerful.
- More then few **optimizations** went in.
  - x4 there: https://github.com/bluealloy/revm/issues/7
- **no_std** from start
- It was **MIT** licence
- I started working on different project but continued working on Revm as a hobby over most of 2022.

2022 - first adopters

- **First adopters:**
  - **Foundry** integrated it 5 months after. https://github.com/foundry-rs/foundry/pull/918
  - **Hardhat** started their own rust backend that was published this year EDR that uses Revm.
  - **Builders/searchers** were first who synced whole mainnet state (And even avalanch).
  - **Helios** light client used it.
- It had amazing traction

End of 2022

- Over the year **all forks** were supported.
- We had **revmjs** a support for javascript, this was abandon later.
- A few more optimizations landed and Revm was very fast:
  - Old benchmark: https://github.com/ziyadedher/evm-bench
- **Reth** started **October 2022**, And i have joined **Paradigm** to help to build it. Have planned to stay only few months but I am still with the project.

2023 - Second year and Reth

- Main focus of the year was on **Reth** client impl.
- **Reth** brought a new user type to Revm (**Chains**).
  - **Optimism** support was added as a feature.
- Support for **Shanghai** and **Cancun** hardforks were added.
- By the end of the year first bigger refactor happened and **Handler** were added. It enabled user to override the Revm logic. It helpe to structure and define components of Evm. More on this later

2024 - a third year

- **Reth 1.0** was release
- **Revm** hit a huge milestone and got **audited**:
  - community-driven and sponsored by six companies that use Revm in various ways
  - Done by Guido Vranken top eth bug hunter
  - https://rakita.github.io/blog/blog/005-revm-audit/
  - In blog post i hinted on possibility of Revm Engame.
- **EOF** got supported
- **zkVM** came into existence and they all use Revm

A critical component of Ethereum Ecosystem.

- This year Revm became one of the most **popular** EVM library and **critical** component in ecosystem.
- Most common **types** of Revm users:
  - **Clients/Chains**: Reth, Helios, Trin, Optimism, Scroll, Bcs, Polygon
  - **Tooling**: Foundry, Hardhat
  - **Builders/Searchers**
  - **zkVM** support for zkEVM: Risc0 Zeth, SP1-reth
    - Formal verification sponsored by EF fondation that targets Revm: https://verified-zkevm.org/
- This affected me to think about **future of Revm.**

Maintenance problems

Problem 1: EIP testing and inclusion

- Example of EIP-1153 https://github.com/bluealloy/revm/issues/154
- **Jul 24, 2022**: Issue is open with **request** to add **EIP-1153**:
  - It was **needed** for test with Foundry. Understandable
  - EIP was **not included** in any fork and I didn't have time to work on it as it was not a priority.
- **Nov 14, 2022** EIP was CFI-ed For **Shanghai** and requested again to be included
  - My response: "As it stands from a few months ago, if you want to invest your own time and experiment, be free to do it in side repo, it is kinda easy to build foundry with patched revm."

Problem 1: EIP testing and inclusion

- **Shanghai** happened and EIP-1153 was **not included**
- **Jul 7, 2023**: PR for EIP was made
- **Aug 3, 2023**: PR was merged
- **March 13, 2024**: EIP shipped with **Cancun**. With work needed to be done **four/five** months **before** for **devnets** and **testnets** forking.

- Another example for EIP-3074: https://github.com/bluealloy/revm/issues/230
- All request were very **reasonable** from **EIP champion** point!

Problem 1: EIP testing and inclusion

Possible paths in this example:

- Should i have started **working** on EIP **right away**?
- If somebody have made PR should i have just **merged** it?
- What to do if **every EIP** champion does this?
- Who is going to **maintain EIPs** that are not included in Spec?

Problem 2: Testing/Dev features

- I try to facilitate all requests that i receive and find the way to make it work.
- Sometimes this is not possible and change directly to code is needed. It is usually done with a feature:, "memory_limit", "optional_balance_check", "optional_block_gas_limit", "optional_eip3607", "optional_gas_refund", "optional_no_base_fee".
- If this was just a Evm, integration with next two project would not be possible:
  - Revmc: https://github.com/paradigmxyz/revmc
  - R55: https://github.com/r55-eth/r55

Problem 3: Chain support

- Most of EVM chains have a small difference with that mainnet Ethereum chain. New tx support, new spec support new EIP support etc..
- Supporting it would require forking of the project
- Over time maintenance cost piles up, new specs needs to be activated and merge conflicts needs to be resolved
- It is error prone and increases risk of consensus bugs.

Solution: EVM Framework Is Revm Engame

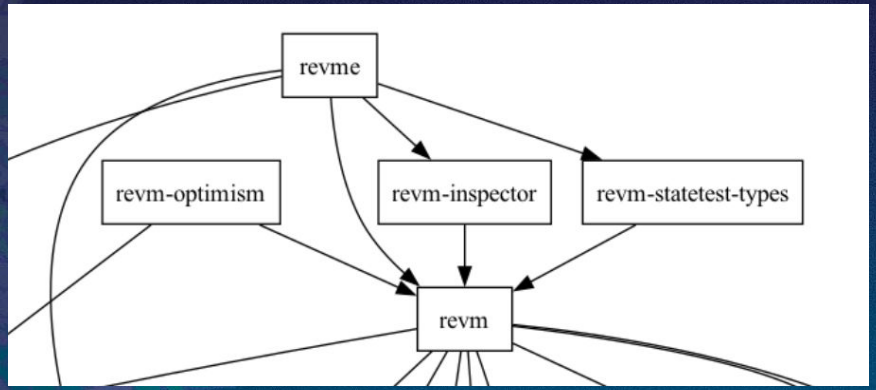
EVM Framework

- Extensible code.
- Chains can use a Revm library without forking.
- Tooling can create custom way to inspect the Evm
- New EIP implementation can be added.
- New execution frames can be supported, revmc/r55/wasm etc
- For example new types could look like:
  - **Main**Evm<DB>
  - **InspectorMain**Evm<DB, INSPECTOR>
  - **Optimism**Evm<DB>
  - **InspectorOptimism**Evm<DB,INSPECTOR>

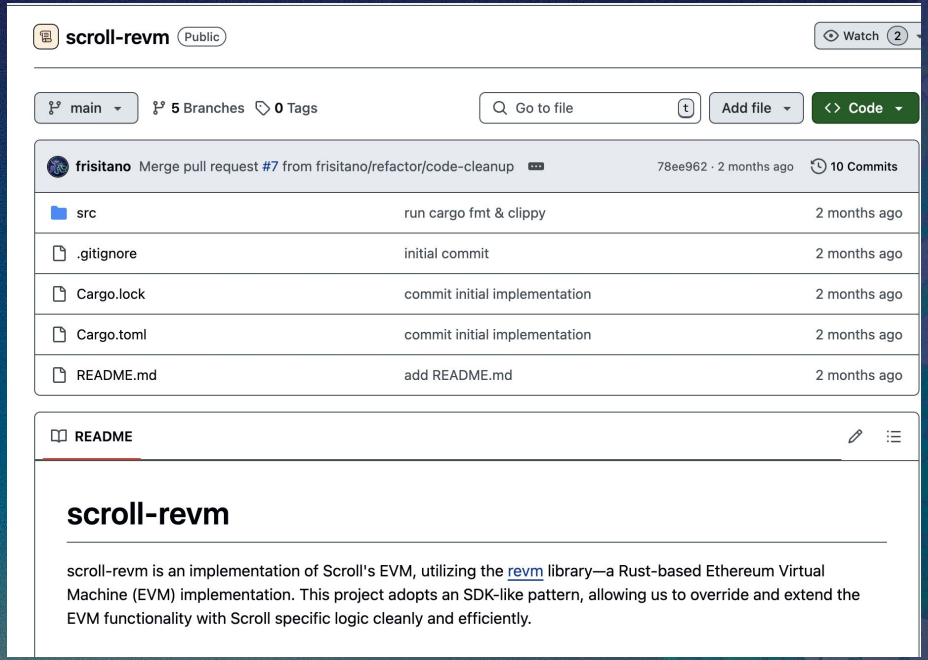
Idea

- **2022** I have looked at **evmone** and noticed that it uses a **array of function** pointers. Was cool idea where there was **multiple array** per fork
- I got **eureka** moment what if we introduce **custom instruction** from outside of repo?
- This ideas was **extended** not just on instruction but on **any logic**. And by end of 2023 **Handler** were made, it was **awesome idea**, but implemented of Box<fn> was **not flexible** and it was **strange** to use and additionally it missed Generic over the data.. **Insight** of data and logic split was made.
- **Evm Framework** introduces those **data generics** and **reworked Handler** into traits for logic.

Examples



- Imagine all EVM chains implementing it in reusable way and tooling using it out of box!



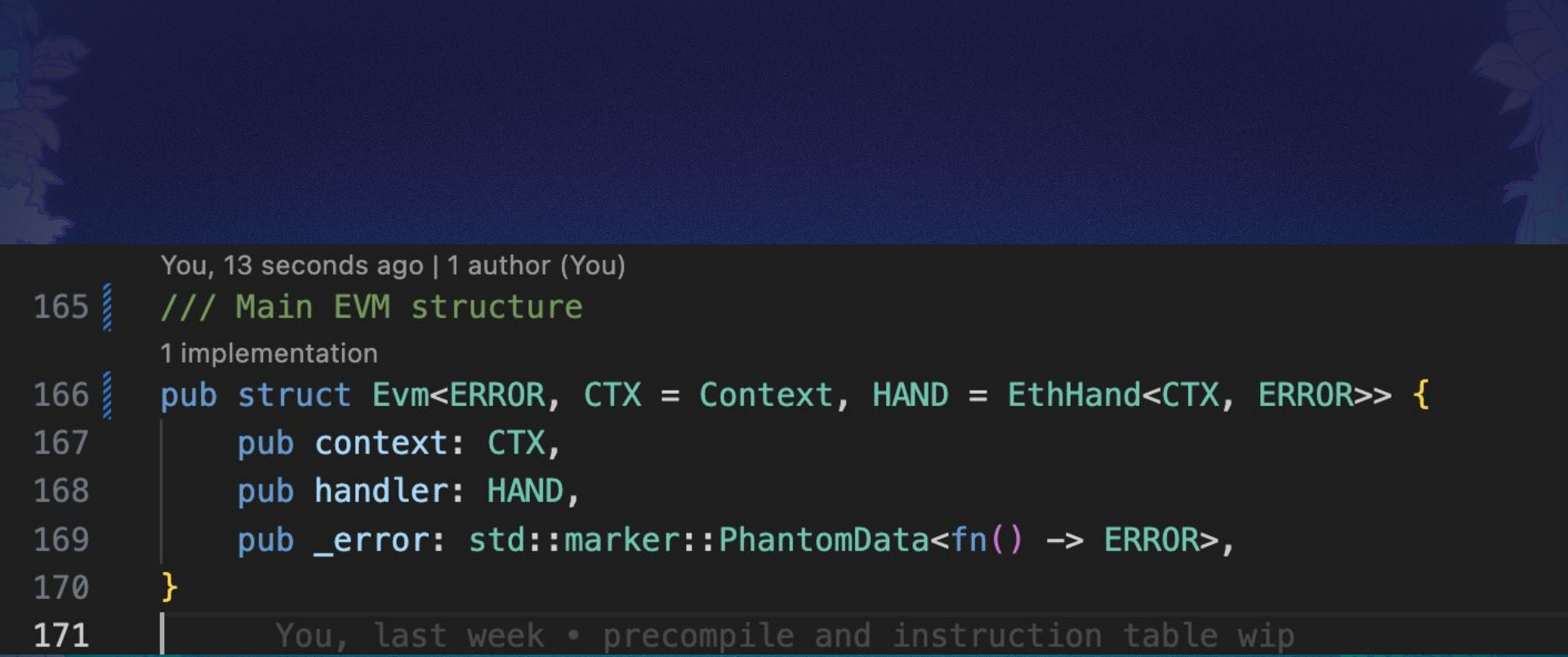
- This is still WIP and new changes are incoming

Split between Data and Logic

- **Context** is the **data**:
  - Transaction
  - Block
  - Cfg
  - Journaled state
    - Database
    - Transient Storage
    - Warming of account
- **Handler** is the **logic** part and it contains:
  - Validation: Validate of Block/Tx/Cfg
  - Pre Execution: Warm Load/Deduct
  - Execution:
    - Frame: Exec loop
      - Interpreter: Interpreter loop
  - Post Execution: Refund/Reimburse/Reward/

```rust
/// EVM context contains data that EVM needs for execution.
#[derive_where(Clone, Debug; BLOCK, SPEC, CHAIN, TX, DB, <DB as Database>::Error)]
14 implementations
pub struct Context<BLOCK = BlockEnv, TX = TxEnv, SPEC = SpecId, DB: Database = EmptyDB, CHAIN = ()>
{
    /// Transaction information.
    pub tx: TX,
    /// Block information.
    pub block: BLOCK,
    /// Configurations.
    pub cfg: CfgEnv,
    /// EVM State with journaling support and database.
    pub journaled_state: JournaledState<DB>,
    /// Inner context.
    pub chain: CHAIN,
    /// TODO include it inside CfgEnv.
    pub spec: SPEC,
    /// Error that happened during execution.
    pub error: Result<(), <DB as Database>::Error>,
}
```

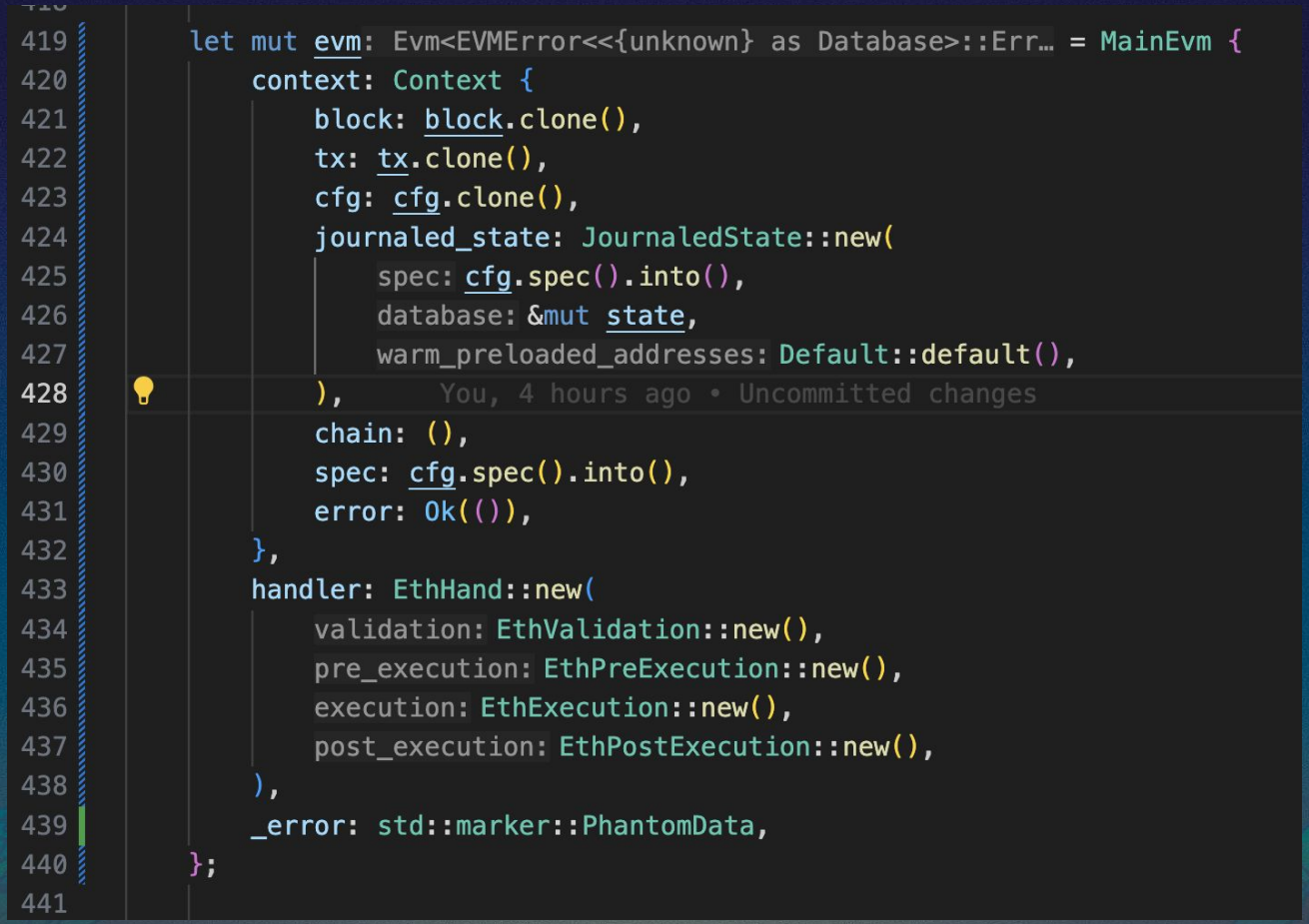
```
You, 4 weeks ago | 1 author (You) | 1 implementation
10  pub trait ValidationWire {
11      type Context;
12      type Error;
13
14      /// Validate env.
15      fn validate_env(&self, context: &Self::Context) -> Result<(), Self::Error>;
16
17      /// Validate transactions against state.
18      fn validate_tx_against_state(&self, context: &mut Self::Context) -> Result<(), Self::Error>;
19
20      /// Validate initial gas.
21      fn validate_initial_tx_gas(&self, context: &Self::Context) -> Result<u64, Self::Error>;
22  }       You, 4 weeks ago • wip of frames
23
```

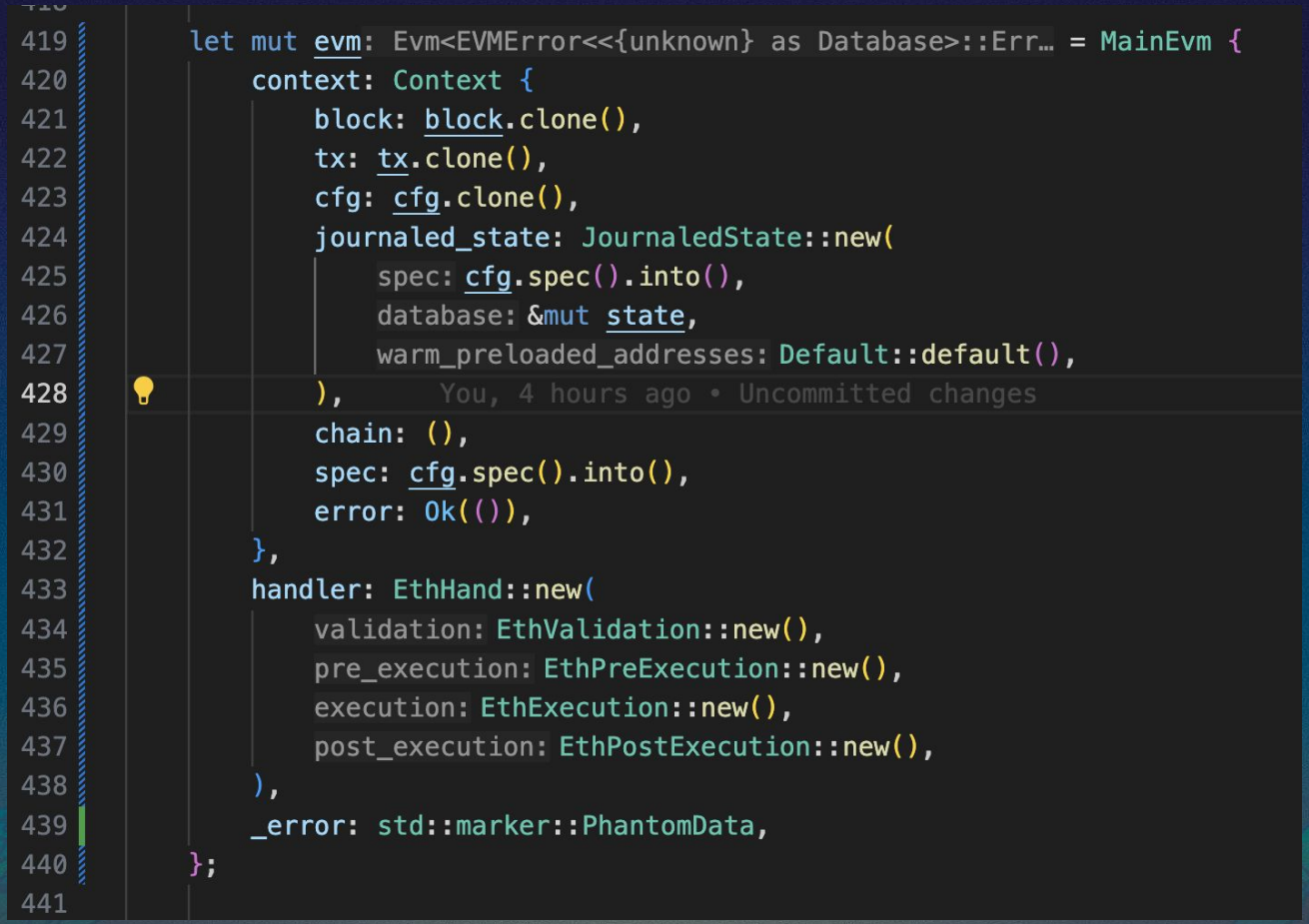

You, 13 seconds ago | 1 author (You)
/// Main EVM structure
1 implementation
pub struct Evm<ERROR, CTX = Context, HAND = EthHand<CTX, ERROR>> {
pub context: CTX,
pub handler: HAND,
pub _error: std::marker::PhantomData<fn() -> ERROR>,
}
You, last week • precompile and instruction table wip

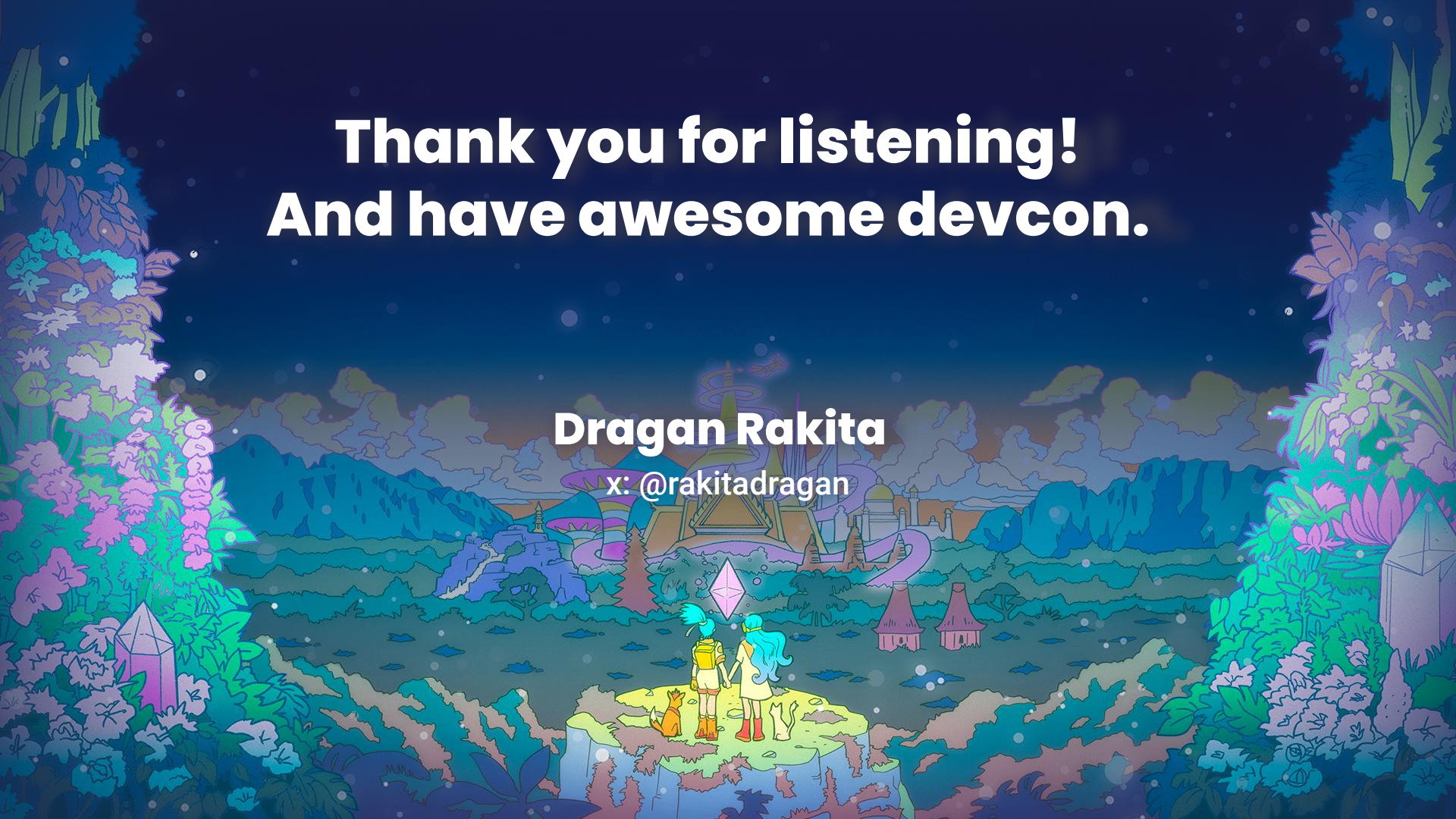
```rust
pub type EthContext<DB> = Context<BlockEnv, TxEnv, SpecId, DB, ()>;

pub type MainEvm<DB> = Evm<
    Error<DB>,
    EthContext<DB>,
    EthHand<
        EthContext<DB>,
        Error<DB>,
        EthValidation<EthContext<DB>, Error<DB>>,
        EthPreExecution<EthContext<DB>, Error<DB>>,
        EthExecution<
            EthContext<DB>,
            Error<DB>,
            EthFrame<
                EthContext<DB>,
                Error<DB>,
                EthInterpreter<()>,
                EthPrecompileProvider<EthContext<DB>, Error<DB>>,
                EthInstructionProvider<EthInterpreter<()>, EthContext<DB>>,
            >,
        >,
    >,
>;
```

```rust
pub type InspCtxType<INSP, DB> = InspectorContext<INSP, BlockEnv, TxEnv, SpecId, DB, ()>;
      You, 7 days ago • Example of inspector in revme
pub type InspectorMainEvm<DB, INSP> = Evm<
    Error<DB>,
    InspCtxType<INSP, DB>,
    EthHand<
        InspCtxType<INSP, DB>,
        Error<DB>,
        EthValidation<InspCtxType<INSP, DB>, Error<DB>>,
        EthPreExecution<InspCtxType<INSP, DB>, Error<DB>>,
        EthExecution<
            InspCtxType<INSP, DB>,
            Error<DB>,
            InspectorEthFrame<
                InspCtxType<INSP, DB>,
                Error<DB>,
                EthPrecompileProvider<InspCtxType<INSP, DB>, Error<DB>>,
            >,
        >,
    >,
>;
```


let mut evm: Evm<EVMError<<{unknown} as Database>::Err… = MainEvm {
context: Context {
block: block.clone(),
tx: tx.clone(),
cfg: cfg.clone(),
journaled_state: JournaledState::new(
spec: cfg.spec().into(),
database: &mut state,
warm_preloaded_addresses: Default::default(),
),
You, 4 hours ago • Uncommitted changes
chain: (),
spec: cfg.spec().into(),
error: Ok(()),
},
handler: EthHand::new(
validation: EthValidation::new(),
pre_execution: EthPreExecution::new(),
execution: EthExecution::new(),
post_execution: EthPostExecution::new(),
),
_error: std::marker::PhantomData,
};


let mut evm: Evm<EVMError<<{unknown} as Database>::Err… = MainEvm {
context: Context {
block: block.clone(),
tx: tx.clone(),
cfg: cfg.clone(),
journaled_state: JournaledState::new(
spec: cfg.spec().into(),
database: &mut state,
warm_preloaded_addresses: Default::default(),
),
You, 4 hours ago • Uncommitted changes
chain: (),
spec: cfg.spec().into(),
error: Ok(()),
},
handler: EthHand::new(
validation: EthValidation::new(),
pre_execution: EthPreExecution::new(),
execution: EthExecution::new(),
post_execution: EthPostExecution::new(),
),
_error: std::marker::PhantomData,
};


Thank you for listening!
And have awesome devcon.
Dragan Rakita
x: @rakitadragan