

The background of the slide is split into two distinct visual fields. On the left, there is a dense, overlapping arrangement of 3D rectangular blocks in various colors including red, orange, yellow, purple, and blue, creating a complex, geometric pattern. On the right, the background is a smooth, out-of-focus gradient transitioning from a warm orange-yellow at the top to a cooler, light purple at the bottom. The title text is positioned on the right side, overlaid on the gradient background.

# Smart Contract With Patract Labs

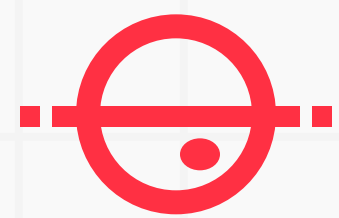
---

*Bonan Yuan*



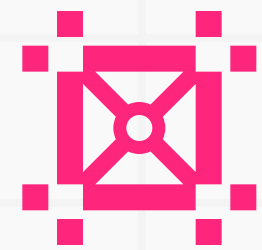
# Who We Are?

Patract Labs provides full-stack support for smart contract development based on Substrate.



## Jupiter

Contract Testnets



## Redspot

Contract Dev Scaffold



## Europa

Contract Sandbox Env



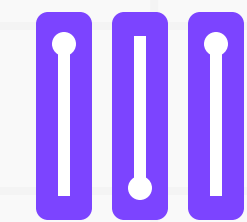
## Parascan

Blockchain Explorer



## Ask!

Assembly Script eDSL for WASM



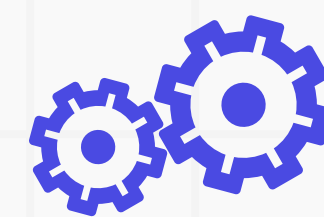
## Metis

Standard Contract Library



## Elara

High Availability Substrate API



## Himalia

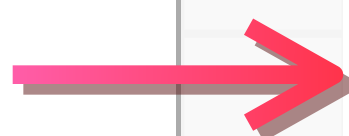
Multi Contract SDKs

# Current Problems With Contract Development

## Our Solution

- **Lack of Multi Language Support.**

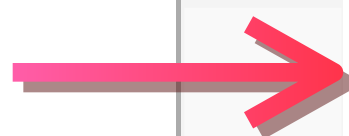
Developers can only program in Rust even WASM supports multi-language compilation.



**Ask!**

- **Lack of Development Suites.**

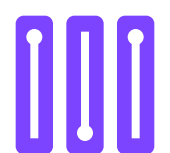
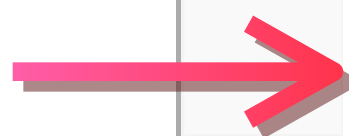
Developers have to deploy and test contract manually.



**Redspot**

- **Lack of Standard Contract Library similar to Openzeppelin.**

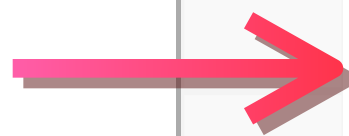
Developers have to manually copy and paste the code, which is time-consuming and error-prone.



**Metis**

- **Lack of Test Sandbox.**

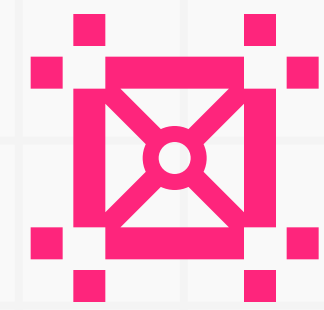
There is no out-of-box contract test sandbox designed specifically for contract development and debugging.



**Europa**

# **Ask!**—Bring AssemblyScript to Substrate

- **eDSL**
  - Similar to ink!, ask! Is eDSL (embedded domain-specific language).
- **Decorators**
  - Using Decorators to interact with `pallet-contracts`
  - High level abstraction, developers can focus on the contract logics.
- **Storage**
  - More storages types: `SpreadStorableMap` `SpreadStorableArray`  
`PackedStorableMap` `PackedStorableArray`
- **Interface and Inheritance**
  - Solidity developers can quick get used to ask!
  - Support Standard Contract Library similar to Openzeppelin.
- **Under development**
  - We are at version v0.2 and will publish v0.3 soon.



# Redspot—Development Suites

- **Template**

- Based on contract standard, automatically generate sample contract, deploy script, test script and Redspot Config.
- `npx redspot-new erc20`

- **Compile**

- Wraps around `cargo-contract`
- `npx redspot compile`

- **Deploy**

- Upload the contract and instantiate it using script.
- `npx redspot run scripts/deploy.ts --no-compile`

- **Test**


- Test contract methods using script.
- `npx redspot test --no-compile`



# Redspot—Flexible and Pluggable

- **Console**
  - Powerful JavaScript Interactive Console for unit testing.
  - `npx redspot console --no-compile`
- **Docker**
  - Test your contract on multiple platforms
  - Start Docker: `$ npx redspot testnet`
  - Compile on Docker: `$ npx redspot compile --docker true`
- **Explorer**
  - Highly integrated into Respot with no more configurations
  - `npx redspot explorer`
- **Plugin**
  - Extend Redspot with custom plugins
  - Stay compatible with Custom Substrate Chain by extending types

# Redspot—Explore



Jupiter A1  
version 2  
#1,183,310

Explorer

Contracts

Accounts

Chain

Tools

JavaScript

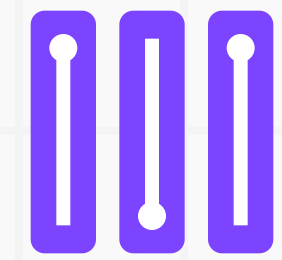
ContractsCodesContractsConsole

+

Add contract bundle

| contract bundles  |                   | status    | upload time   |
|---|-------------------|-----------|---------------|
| <div><div>&lt;/&gt;</div>ERC20-E1134C</div> <div><div>Constructors (1)</div><div><div>deploy</div><div><div>new (initialSupply: <i>Balance</i>)</div><div>Creates a new ERC-20 contract with the specified initial supply.</div></div></div><div><div>Messages (6)</div><div>36,488 WASM bytes</div></div></div> <td>0xe1134c...0f3041</td> <td>Available</td> <td>8 minutes ago</td> | 0xe1134c...0f3041 | Available | 8 minutes ago |





# Metis—Rusty Solution For Standard Contract Library

## Current problems when developing in ink!

- **Lack of standard library makes the smart contract unsafe.**
  - The DAO attack in 2016 stole 3.6m Ether due to a bug in the smart contract. This attack enforced hardfork for Ethereum Mainnet.
- **Developers have to manually copy/paste existing implementation.**
  - The source code could be unaudited and unsafe to use.
  - Time consuming and error-prone during the process.

## Metis

- **Standard Contract library based on reusable components.**
  - Unlike Openzeppelin's Inheritance model, Metis is based on Components, which means users do not directly inherit the standard implementation. Instead, Metis provides a set of reusable components for users to assemble.



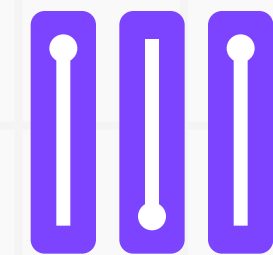
# **Metis**—Composition, not inheritance

## **Inheritance** — *Used by Openzeppelin*

- **Pros:**
  - **Simplicity** : Minimize the code to write for developers.
- **Cons:**
  - **Ambiguity** : Conceal method definitions; Uncertain Inheritance tree with multiple Inheritance.

## **Composition** — *Used by Metis*

- **Pros:**
  - **Clarity** : Improve code readability and audibility.
- **Cons:**
  - **Repetition** : Repeat writing the same code for existing implementation.



# Metis—MCCI architecture

**M**

*Data model*

- Contract State mapping.
- One data model per component.

**C**

*Component*

- Reusable.
- Encapsulation of data and methods.
- Independent.
- Orthogonality with other components.

**C**

*Controller*

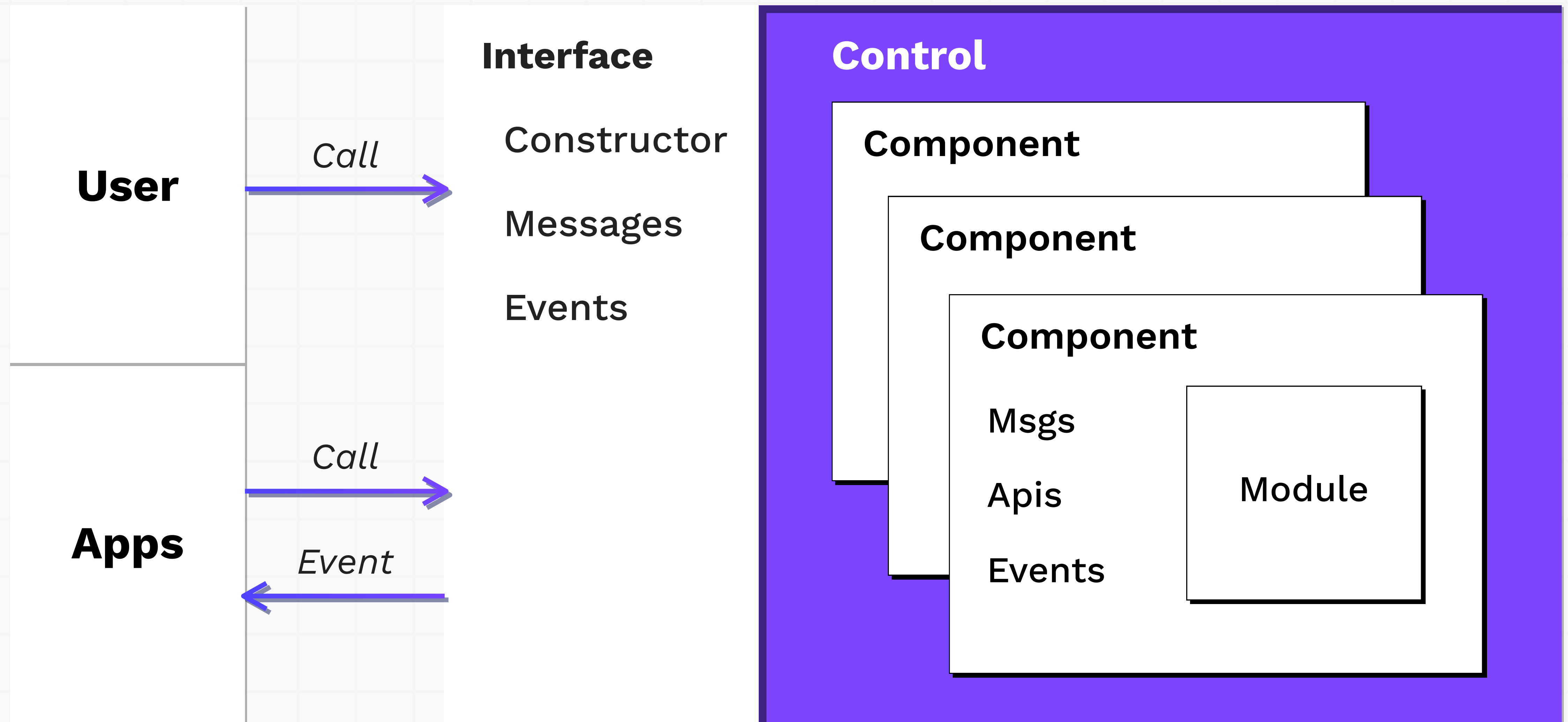
- Coordinates the components and implements the contract interface.

**I**

*Interface*

- The interface is the user interface of the contract.
- The interface defines the behavior of the contract and, to some extent, defines metadata.

# Metis—MCCI architecture





# Metis—vs Native ink! (Storage)

**Standardize Storage structure for ERC20 token**

ink!

Metis



```
#[ink(storage)]
#[import(erc20)]
pub struct Erc20 {
    erc20: erc20::Data<Erc20>,
}
```



```
#[ink(storage)]
pub struct Erc20 {
    /// Total token supply.
    total_supply: Lazy<Balance>,
    /// Mapping from owner to number of owned token.
    balances: StorageHashMap<AccountId, Balance>,
    /// Mapping of the token amount which an account is allowed to
    withdraw
    /// from another account.
    allowances: StorageHashMap<(AccountId, AccountId), Balance>,
}
```

# Metis—vs Native ink! (Constructor)

**Skip the process of variable declaration; No need to manually emit event**

## Metis

```

● ● ●

#[ink(constructor)]
pub fn new(initial_supply: Balance) -> Self {
    let mut instance = Self {
        erc20: erc20::Data::new(),
    };

    erc20::Impl::init(
        &mut instance,
        String::from("MetisTestToken"),
        String::from("MET"),
        18_u8,
        initial_supply,
    );

    instance
}

```

## ink!

```

● ● ●

#[ink(constructor)]
pub fn new(initial_supply: Balance) -> Self {
    let caller = Self::env().caller();
    let mut balances = StorageHashMap::new();
    balances.insert(caller, initial_supply);
    let instance = Self {
        total_supply: Lazy::new(initial_supply),
        balances,
        allowances: StorageHashMap::new(),
    };
    Self::env().emit_event(Transfer {
        from: None,
        to: Some(caller),
        value: initial_supply,
    });
    instance
}

```



# Metis—vs Native ink! (Event)

Stays pretty much the same due to current design of Ink!. Will mitigate in near future

## Metis



```
#[ink(event)]
#[metis(erc20)]
pub struct Transfer {
    #[ink(topic)]
    from: Option<AccountId>,
    #[ink(topic)]
    to: Option<AccountId>,
    value: Balance,
}
```

## ink!



```
#[ink(event)]
pub struct Transfer {
    #[ink(topic)]
    from: Option<AccountId>,
    #[ink(topic)]
    to: Option<AccountId>,
    value: Balance,
}
```



# Metis—vs Native ink! (Message)

ink!

**Contains default method implementation that ensures the security of balance transfer**

## Metis

```

● ● ●

#[ink(message)]
pub fn transfer_from(
    &mut self,
    from: AccountId,
    to: AccountId,
    value: Balance,
) -> Result<()> {
    erc20::Impl::transfer_from(self, from, to, value)
}

```

```

● ● ●

#[ink(message)]
pub fn transfer_from(
    &mut self,
    from: AccountId,
    to: AccountId,
    value: Balance,
) -> Result<()> {
    let caller = self.env().caller();
    let allowance = self.allowance(from, caller);
    if allowance < value {
        return Err(Error::InsufficientAllowance)
    }
    self.transfer_from_to(from, to, value)?;
    self.allowances.insert((from, caller), allowance - value);
    Ok(())
}

```

# Metis—Design Principle

## Why Can't we design the library similar to the right side?

- Instantiation means create a instance of smart contract on the blockchain.
- Calling method of another instance means calling method from another contract.
- Cross contract calling ends up with wrong result.

## Desired Implementation (pseudocode):

```

contract MyErc20 {
    ERC20 myerc20 = erc20.new(some params);
    fn mint(int amount){
        self.myerc20.mint(amount)
    }
    fn balance(){
        self.myerc20.balance()
    }
}

```



# Metis

## Composition Of Multiple Components



```
#[ink(storage)]
#[import(erc20, ownable, pausable)]
pub struct Erc20Pausable {
    erc20: erc20::Data<Erc20Pausable>,
    ownable: ownable::Data<Erc20Pausable>,
    pausable: pausable::Data,
}
```

### ERC20 Pausable - Storage



```
impl Erc20Pausable {
    #[ink(constructor)]
    pub fn new(initial_supply: Balance) -> Self {
        let mut instance = Self { erc20: erc20::Data::new(), ownable:
ownable::Data::new(), pausable: pausable::Data::new(), };
        erc20::Impl::init( &mut instance,
String::from("MetisTestToken"),
        String::from("MET"), 18_u8, initial_supply, );
        ownable::Impl::init(&mut instance);
        pausable::Impl::init(&mut instance);
        instance
    }
}
```

### ERC20 Pausable - Constructor





# Europa—Out-of-box Local Test Net

- **Another Substrate Implementation** *Focus on smart contract development*
- **No consensus** *Only produce blocks when receiving extrinsics*
- **No WASM Runtime**
- **State KV Database** *Trace block state changes*
- **Contract Pallet Modification** *Reveal the blackbox of contract execution*



# Europa UI—One Click Substrate node with GUI

- Europa UI is specially designed for contract development.
- Download the binary release and start your own node with one click
- Available on all platforms



Europa is Ready!

Database Path ⓘ



Workspace ⓘ

HTTP Port

WS Port

Redspot Projects

 Add  Delete

☒ More Options

Start



# Europa UI

## Contract Pages

Contract > Instances > erc20

Search by Txn hash/Block

Address

5DzVoGuijVCDqZumHFw5PSzRUH6Txpi1HrSubxSyXCZ2...

Creator

5GrwvaEF5zXb... at 0x93fd198542...

Balance

9.9994 DOT

Functions

Extrinsics

Events

Read

Execute

1

totalSupply

2

balanceOf

owner: AccountId

alice : 5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY

Read

Call With Trace

From 5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY

Value 0

To erc20 : 5DzVoGuijVCDqZumHFw5PSzRUH6Txpi1HrSubxSyXCZ2MhWG

Gas Limit 3617419000

Gas Consumed 3616533053

Gas Left 885947

Function

balance\_of

Trap Reason

{  
 "Return": {  
 "data": "0x00000000000000000000000000000000",  
 "flags": 0  
 }  
}

Args

|       |  |
|-------|--|
| owner | 5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY |
|-------|--|

Env trace

|                      |     |  |
|----------------------|-----|--|
| SealValueTransferred | out | 0  |
| SealInput            | buf | 0x0f755a56d43593c715fdd31c61141abd04a99fd6822c8558854ccde39a5684e7a56da27d |

## Nested View of Cross Contract Call

From 5GrwvaEF5zXb26Fz9rcQpDWS57CtERHpNehXCPcNoHGKutQY

Value 0

To all : 5EQLteyeuTER3hyDoPJGWgSn2hCrNH08MY93xEgyYk7p3k9j

Gas Limit 400000000000

Gas Consumed 10409671053

Gas Left 389590328947

More Details

From 5EQLteyeuTER3hyDoPJGWgSn2hCrNH08MY93xEgyYk7p3k9j

Value 0

To erc20 : 5D1eqffigDNMmd2Gfrmo1Jh6gKiC1CxeEZ1njJnTwEmg7dtW

Gas Limit 396923209666

Gas Consumed 2508918335

Gas Left 394414291331

More Details

From 5EQLteyeuTER3hyDoPJGWgSn2hCrNH08MY93xEgyYk7p3k9j

Value 0

To miner\_erc20 : 5FdGbzQsZukbbSN1Wph7v5cVIY5cWFaMjCXTh1RgJWNv...

Gas Limit 392662790687

Gas Consumed 3214283795

Gas Left 389448506892

More Details





# Europa CLI—No More Blackbox For Contract Execution

## Custom RPCs and Commands

- **RPCs**

europa\_forwardToHeight

europa\_backwardToHeight

europa\_modifiedStateKvs

- **Commands**

State-kv

Workspace

## Detailed Logging for Contract Execution

- **Trace all parameters and blockchain changes during contract execution**

- **WASM panic backtrace**

# Demo

**1** Use Redspot to quickly setup development environment with built-in templates

**3** Showcase writing an ERC20 Pausable Smart contract with Access Control

**2** Start Europa UI

**4** Compile and deploy the smart contract using Redspot

**5** Test the smart contract methods using Europa

The background is a vibrant, abstract composition of overlapping geometric shapes, primarily rectangles and parallelograms, in a warm color palette of yellows, oranges, reds, and purples. The shapes are arranged in a way that creates a sense of depth and movement, with some elements appearing to recede into the background while others come forward. The overall effect is a dynamic and visually rich pattern.

**Thanks for watching**