RDX WORKS

Why DeFi & Web3.0 are the Future

(on Radix)

Point #1

DeFi/Web3.0 allows any developer to build an app that can do **anything** done in traditional finance today, but *faster*, *cheaper*, and *better*

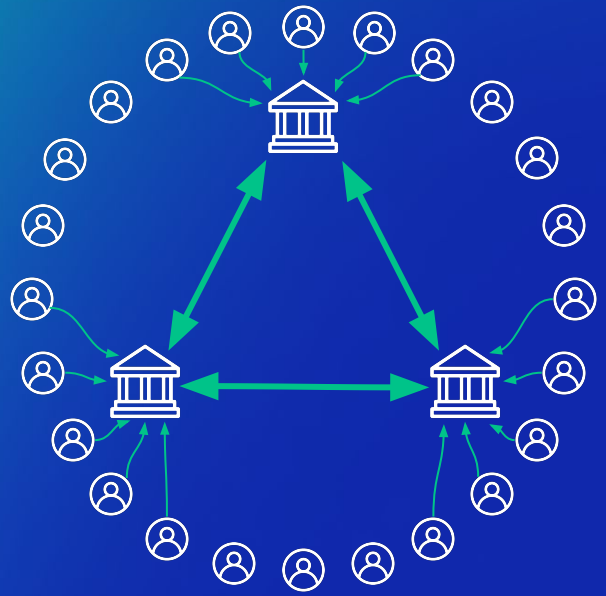
... plus things that traditional finance and Web2.0 could never do.

RDX WORKS

Photo by Bill Jelen on Unsplash

Traditional Finance

Closed network(s)
Institutional control of assets
Institutions create all applications

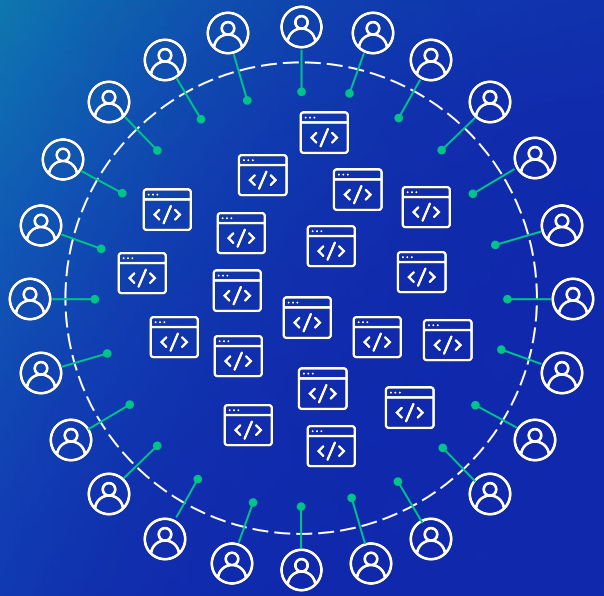


Financial products and services

RDX WORKS

The DeFi/Web3.0 Vision

Open network
Individual control of assets
Individuals create all applications

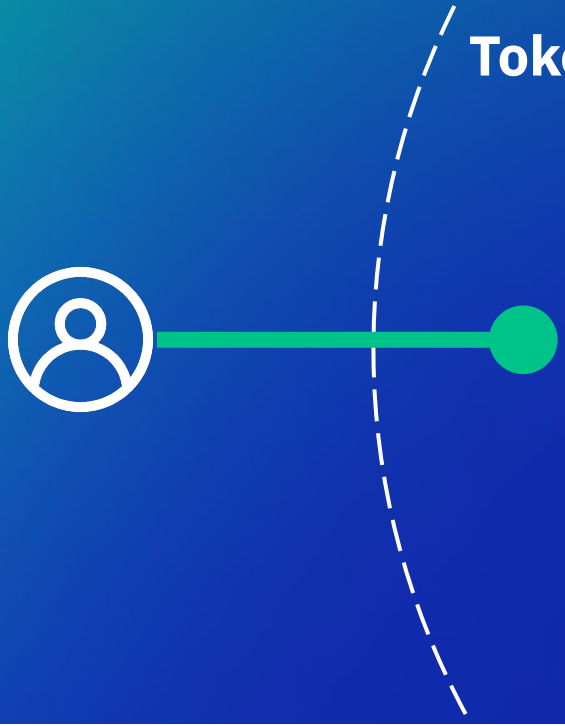


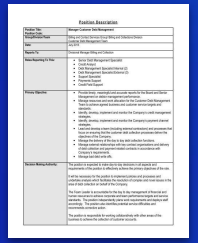
1000s of finance dApps
True asset ownership on the web
True identity ownership on the web

RDX WORKS

Superpower 1

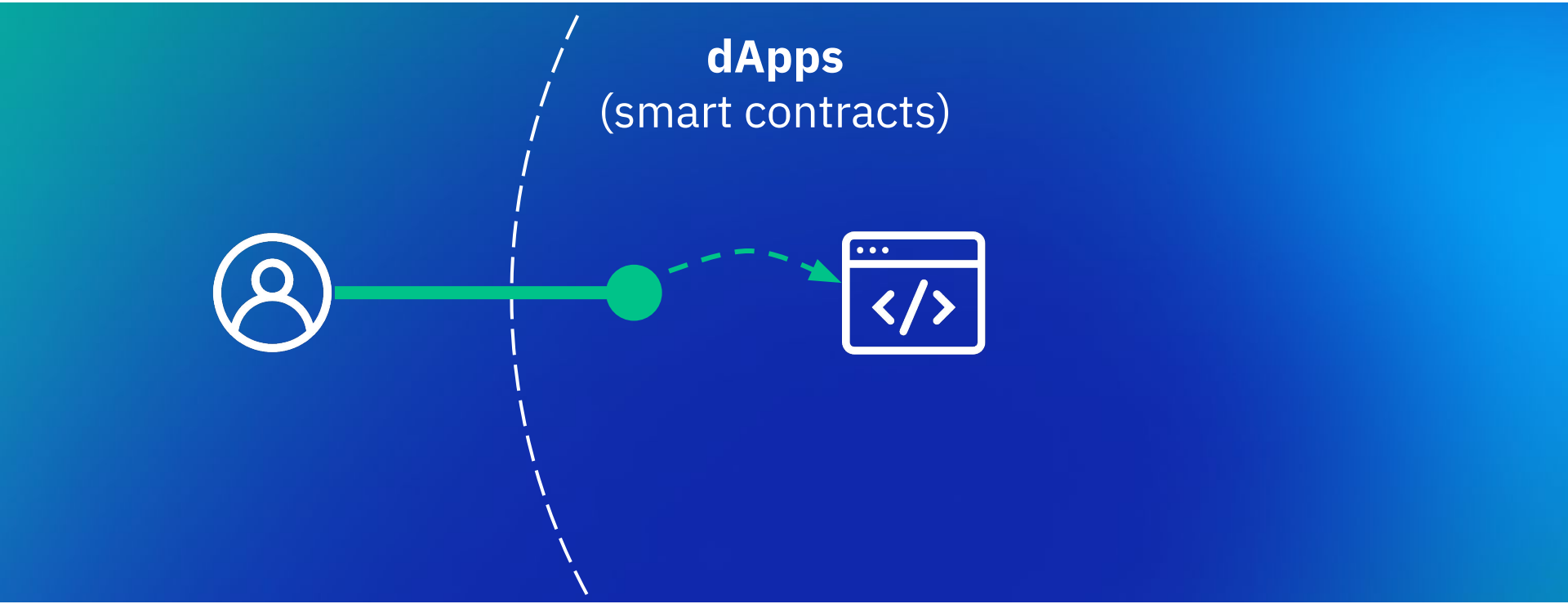
Tokenized Assets





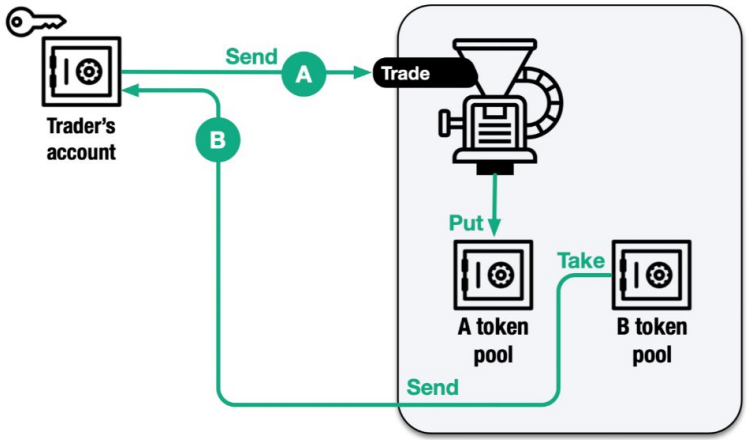
RDX WORKS

Superpower 2



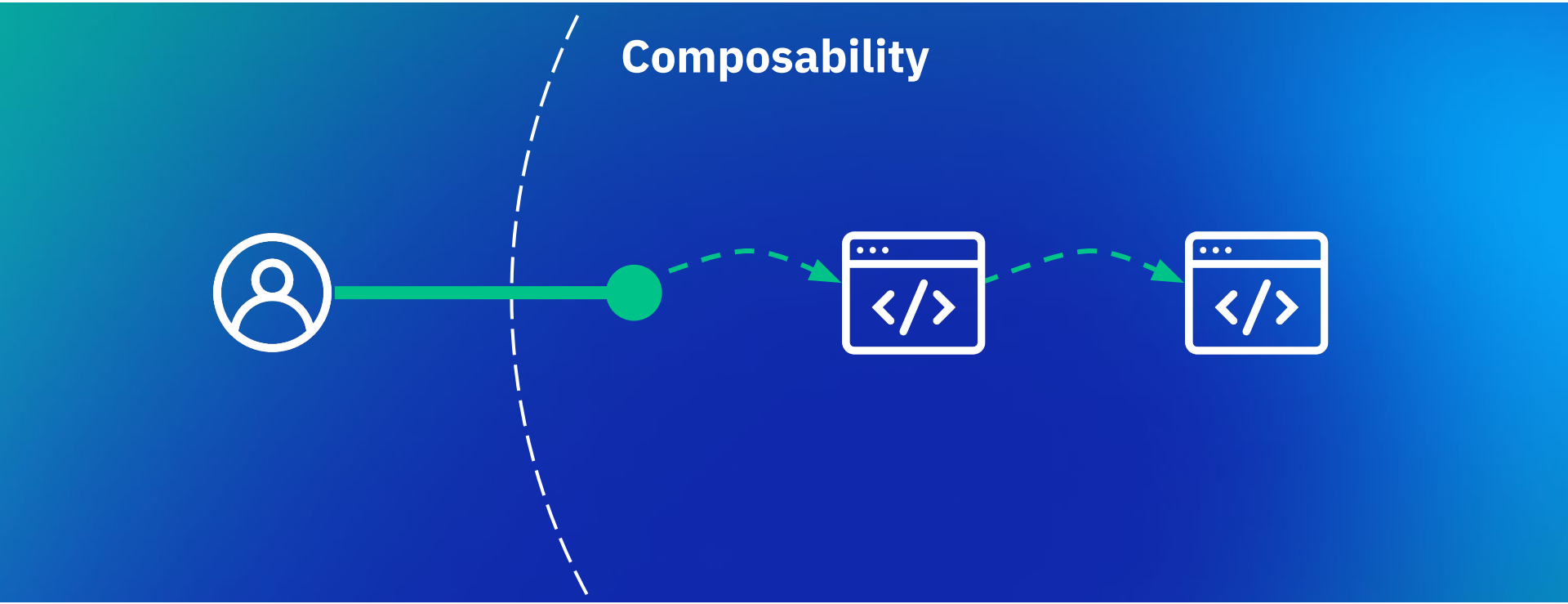
RDX WORKS


Currency Exchange


Send
A
Trade
Trader's account
B
Put
Take
A token pool
B token pool
Send

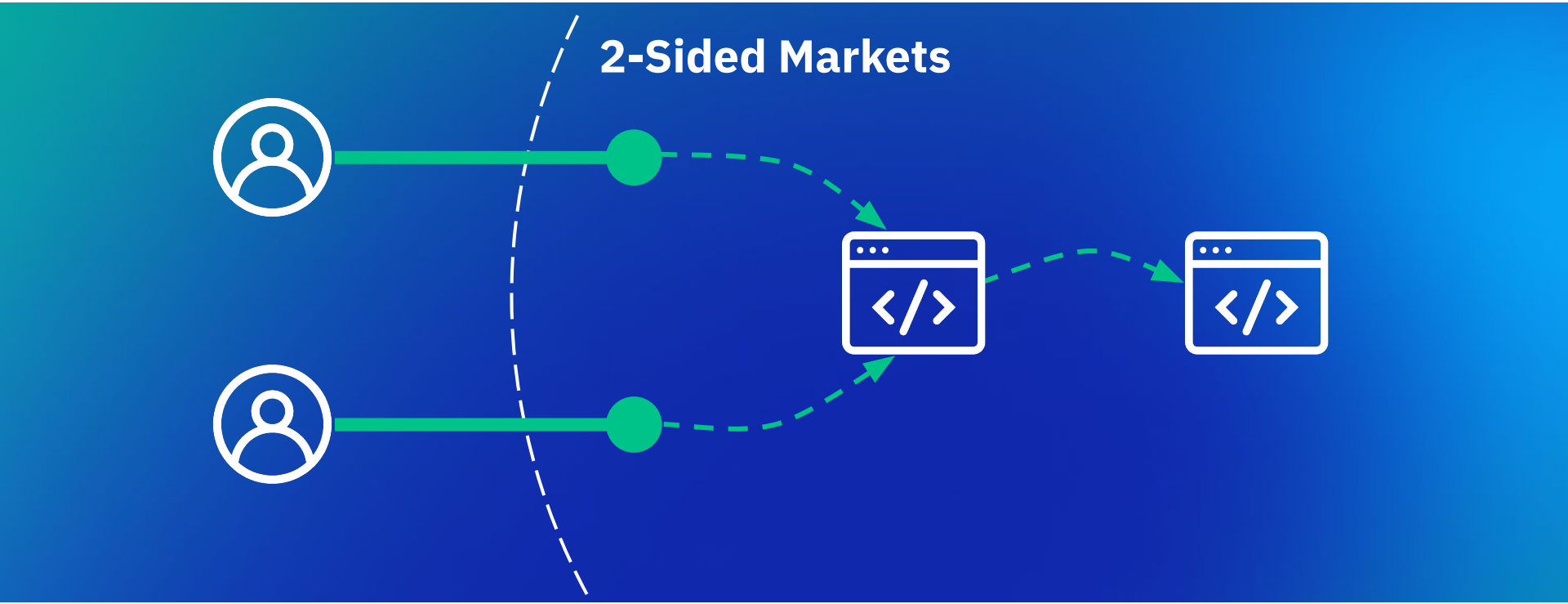
RDX WORKS

Superpower 3



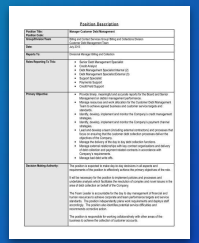
RDX WORKS

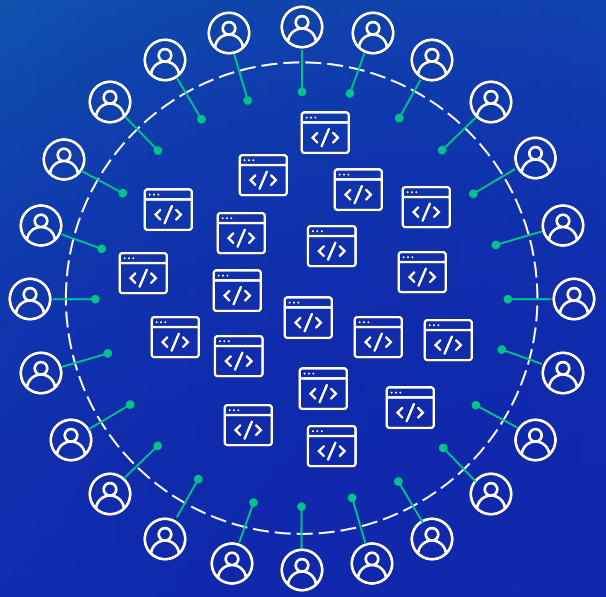
Superpower 4



RDX WORKS


THE UNITED STATES OF AMERICA
ONE DOLLAR






Currency
US Dollar
USA
Singapore Dollar
Singapore
日本円 (:100)
Japan
人民币
China
Bank Buys Notes
31.51
23.46
25.83

Global Finance

DeFi Now
(June 2022)

$0.1 TN

RDX WORKS

Point #2

The opportunity for developers in DeFi/Web3.0 is **tremendous.**

But current platforms are difficult to use and limited in capability.

RDX WORKS

Builder experience is terrible

Few dApps reach production state.
dApps are often exploited.

Platforms are limited in capacity

Transaction fees are often high.
Successful dApps reach limits.

RDX WORKS

Built for Builders

Built for Scale

Scrypto + Radix Engine

Asset-oriented smart contracts

Fastest path from idea to production DeFi dApp.

Cerberus

“Braided” cross-shard consensus

Unlimited dApp throughput. Low fees. Forever.

RDX WORKS

RDX WORKS

Introducing Scrypto
Asset-oriented concepts

Presentation Overview

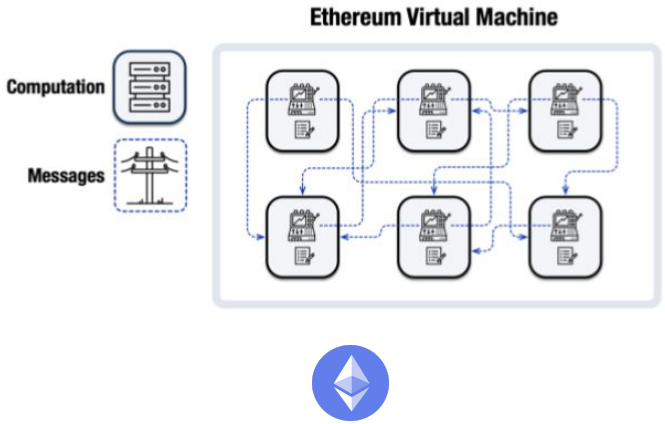
In this Presentation:

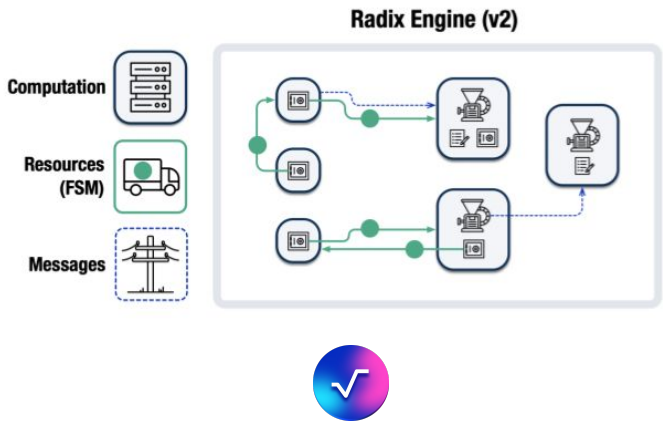
- **Resources**: Tokens, NFTs, and other assets
- **Components and Blueprints:** Radix's form of "smart contracts"
- **Accounts and Transactions**
- **Authorization:** Doing authorization through badges and resources.

Resources

Tokens, NFTs, and other assets

RDX WORKS

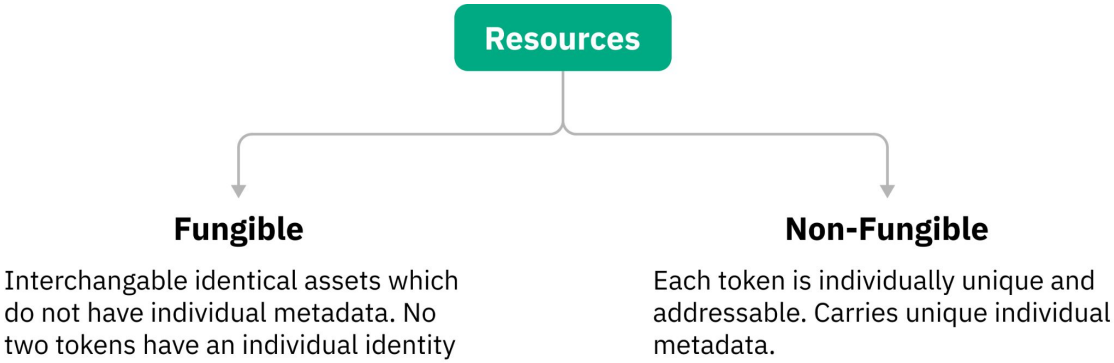

Ethereum Virtual Machine
Computation
Messages


Radix Engine (v2)
Computation
Resources
(FSM)
Messages

Resources

What is a Resource?

- Resources are a **native** implementation of assets on the Radix platform.
- **Resources** are **understood** by the platform and it knows how to deal with them.
- Resources are implemented through an **FSM** which guarantees their behavior.



Resource Containers

Where do Resources Live?

Resources must always exist in a **Bucket** or **Vault**. They share a lot in common:

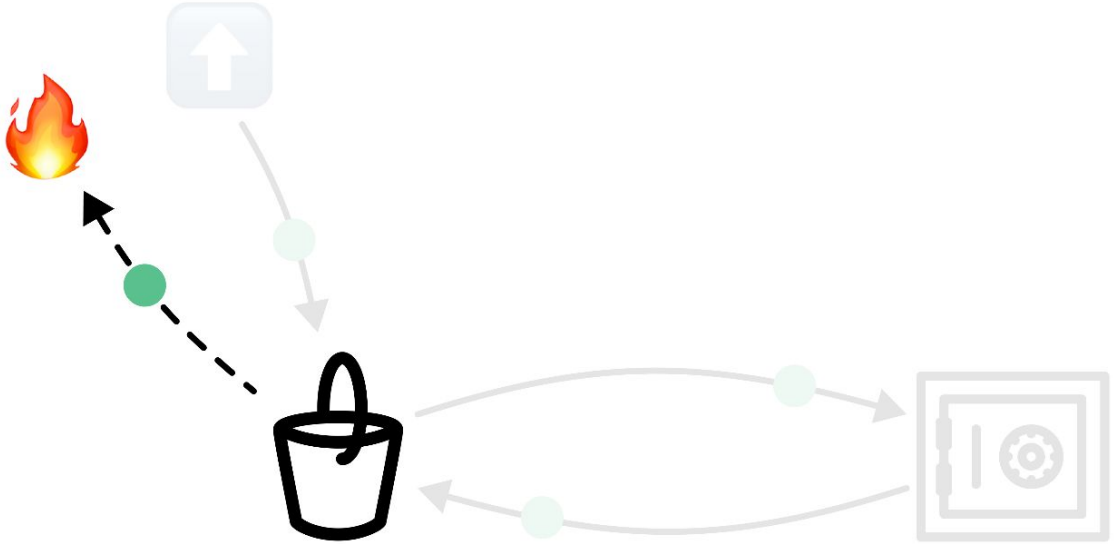
- Both are resource containers.
- Both can only hold one resource address.

Buckets

Temporary resource container. When a transaction ends, no buckets can remain.

Vaults

Permanent resource container. Stored inside components*. Can not be moved around.



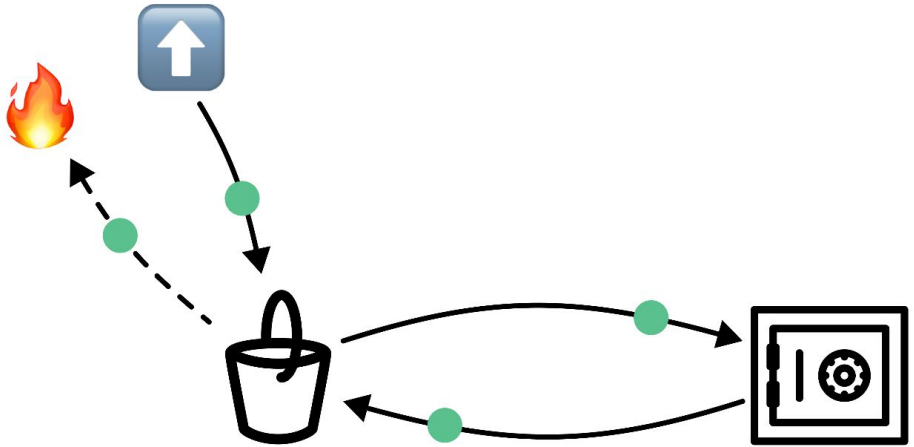
Resource is Burned

Burning of resources is another transition on the finite state machine. To modify the behavior of tokens, we can control who can perform the state transitions.

Resource Finite State Machine

Why a Finite State Machine model for Resources?

- Makes the idea of tokens very intuitive and easy to understand and reason about.
- Resources are more reliable.
- States which have not been enumerated are impossible.
- Higher **guarantees** and more **transparency**.



Controlling Resource Behavior

Resource behavior is controlled by controlling who can perform the state transitions

- mintable
- burnable
- restrict_withdraw
- restrict_deposit
- updateable_metadata
- recallable

Resource Creation

Resource Creation

When resources are first created, they go into a bucket

Resource Types

There are two resource types which you specify on the `ResourceBuilder`: `fungible` and `non_fungible`.

```
1  let bucket: Bucket = ResourceBuilder::new_fungible()
2      .metadata("name", "Company Shares")
3      .metadata("symbol", "SHARE")
4      .metadata("description", "Each token represents a share in the company.")
5      .mintable(rule!(require(admin_badge)), LOCKED)
6      .burnable(rule!(allow_all), LOCKED)
7      .initial_supply(1_000);
```

State Transitions

Defines the rules of the state transition in the resource finite state machine.

Components and Blueprints

the Radix form of “smart contracts”

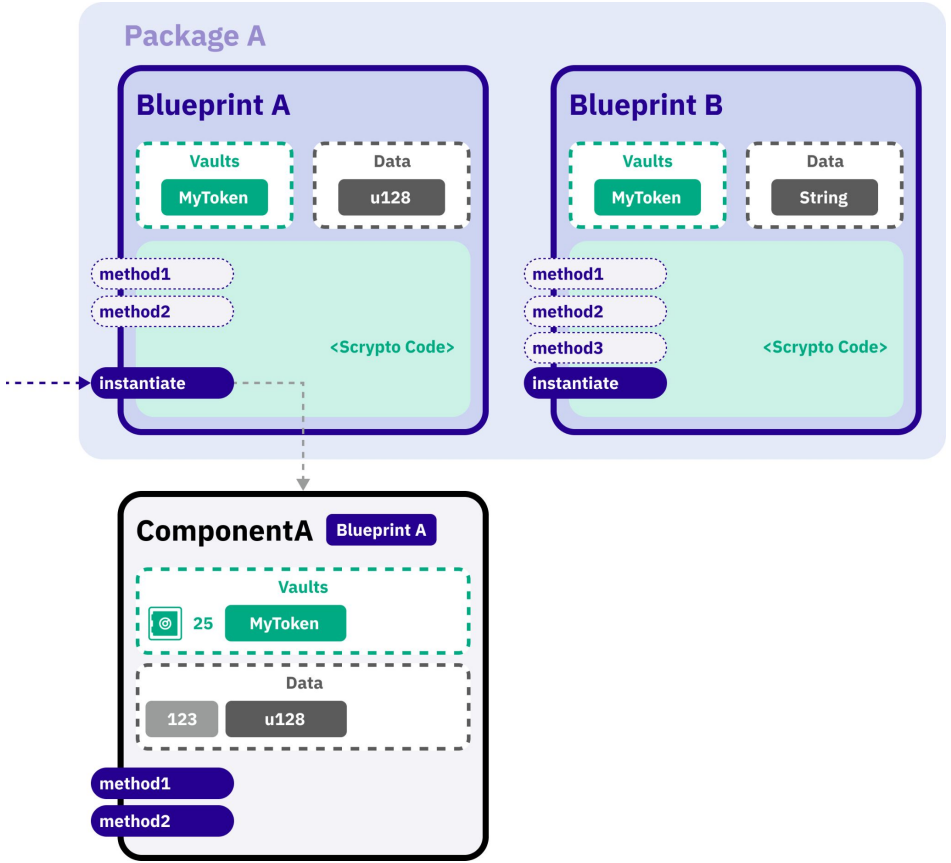
RDX WORKS

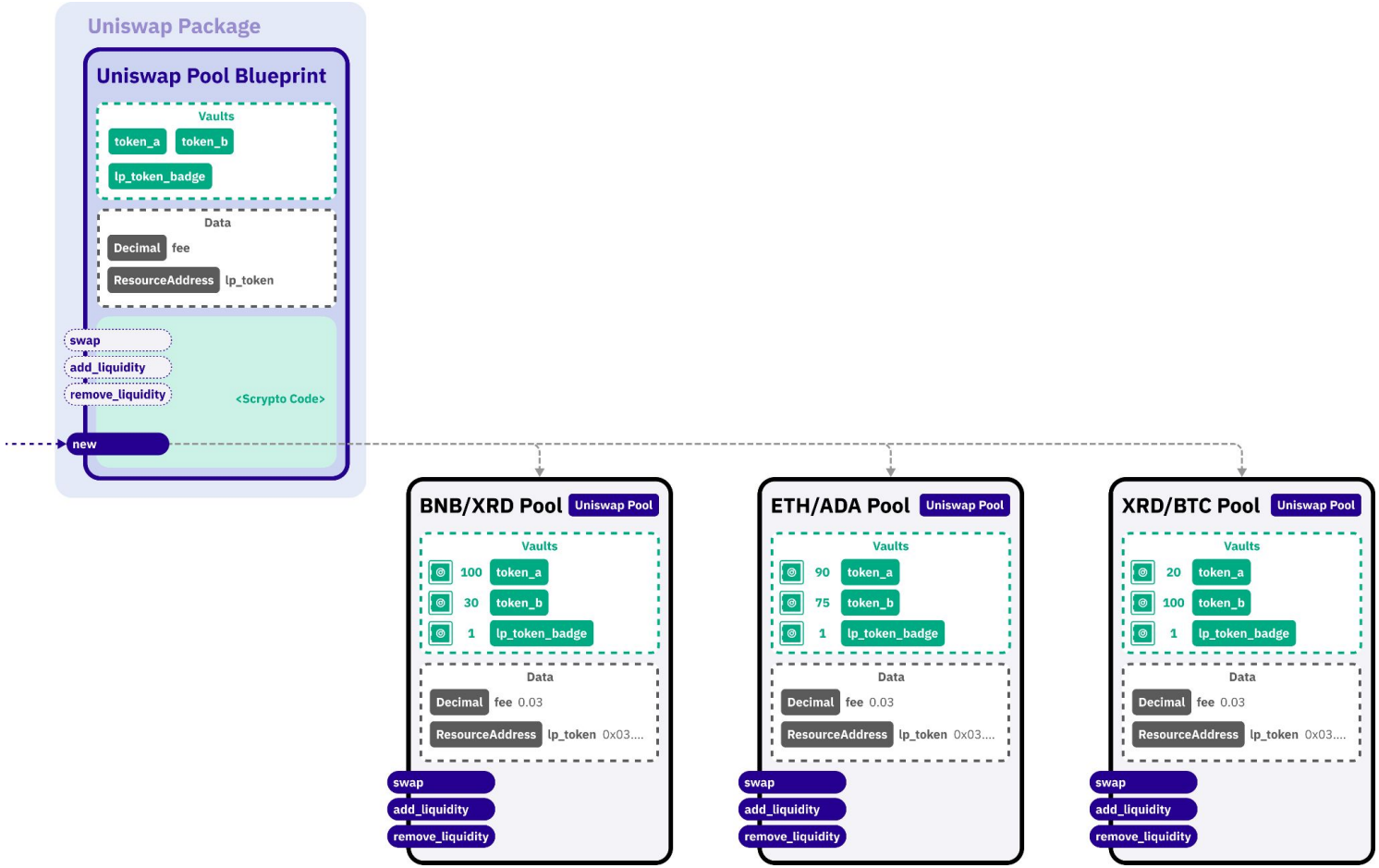
Components

Component: An instance of a blueprint. Has an address and maintains state.

Blueprint: Defines a shared structure and behavior. Does not have an address and can not hold state.

Package: A collection of blueprints compiled and published as as single unit. Has an address.

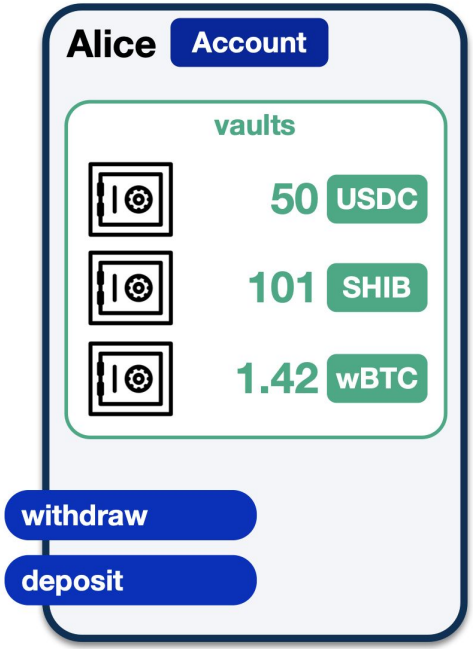



Uniswap Package
Uniswap Pool Blueprint
Vaults
token_a
token_b
lp_token_badge
Data
Decimal
fee
ResourceAddress
lp_token
swap
add_liquidity
remove_liquidity
<Scrypto Code>
new
BNB/XRD Pool
Uniswap Pool
Vaults
100
token_a
30
token_b
1
lp_token_badge
Data
Decimal
fee 0.03
ResourceAddress
lp_token 0x03....
swap
add_liquidity
remove_liquidity
ETH/ADA Pool
Uniswap Pool
Vaults
90
token_a
75
token_b
1
lp_token_badge
Data
Decimal
fee 0.03
ResourceAddress
lp_token 0x03....
swap
add_liquidity
remove_liquidity
XRD/BTC Pool
Uniswap Pool
Vaults
20
token_a
100
token_b
1
lp_token_badge
Data
Decimal
fee 0.03
ResourceAddress
lp_token 0x03....
swap
add_liquidity
remove_liquidity

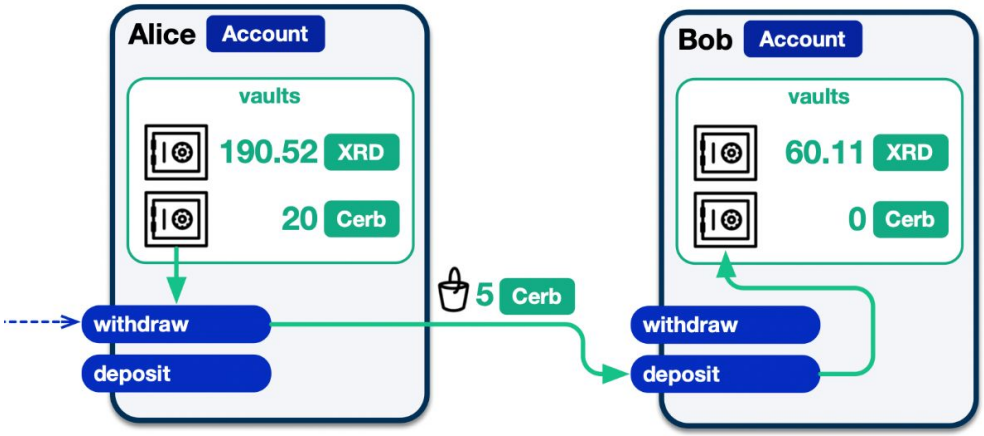
Accounts and Transactions

RDX WORKS

Accounts


Alice
Account
vaults
50
USDC
101
SHIB
1.42
wBTC
withdraw
deposit

Transactions



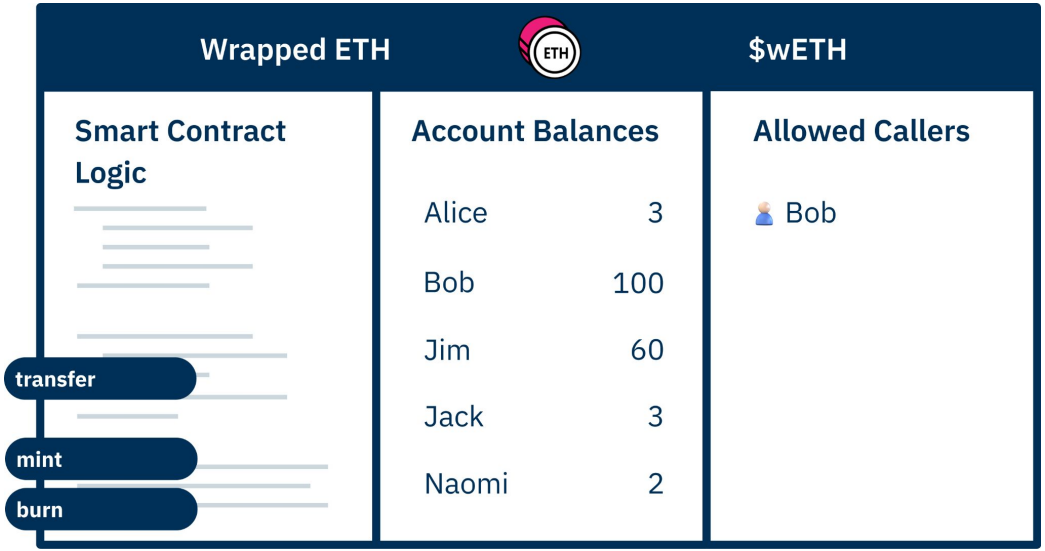
Authorization

RDX WORKS

Authorization through the Caller Address

We want to only allow Bob to call this smart contract.

"Ah! Perhaps I can tell my contract to only accept calls from Bob's Address!"

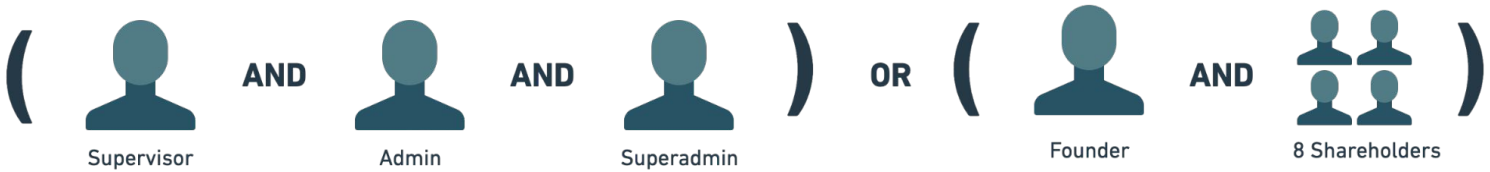


RADIX

Authorization through the Caller Address

Address-based authentication has many downsides:

- Can **lead to hacks** when caller address manipulation is possible.
- Assumes a one to one mapping of public keys to accounts.
- Modeling **authorization** rules is ***very*** **complex**.
- Additional functions are required for **auth hand-off**.

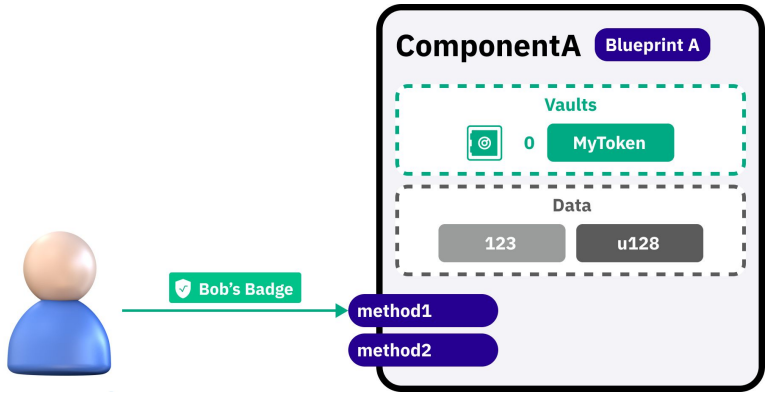


RADIX

Authorization in Scrypto

Authorization in Scrypto is based on ***what you have* instead of *who you are***.

- Allows out-of-the-box support for **role-based authorization**.
- **Does not require additional methods** for authorization hand-off.
- Very well integrated with Scrypto, Makes the modeling of **complex auth very simple**.



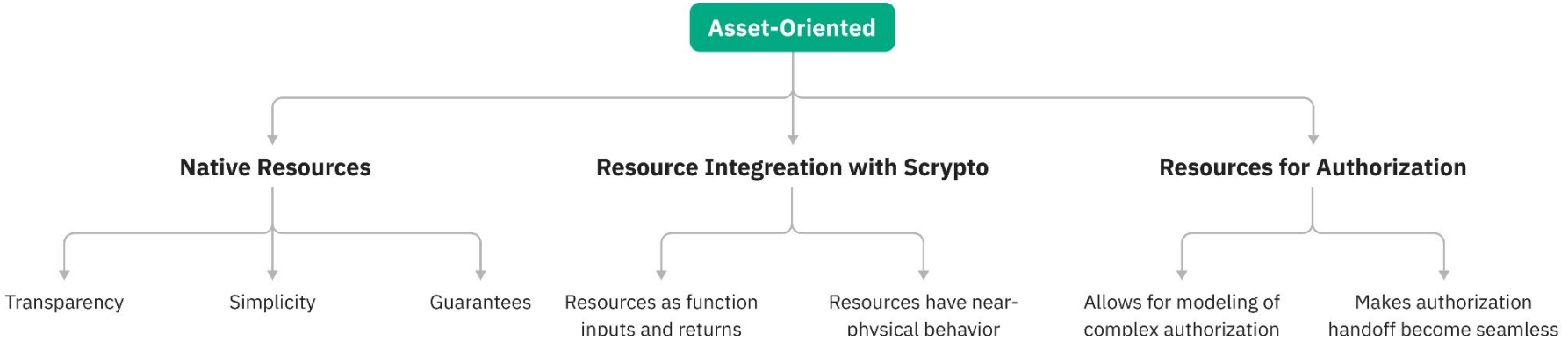
RADIX

Authorization in Scrypto

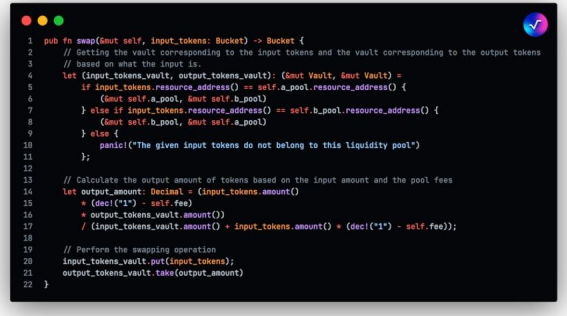

```
let access_rule: AccessRule = rule!(
    (require(supervisor_badge) && require(admin_badge) && require(superadmin_badge))
    || (require(founder_badge) && require_amount(8, shareholder_badge))
);
```

RADIX

What does Asset Oriented Mean?



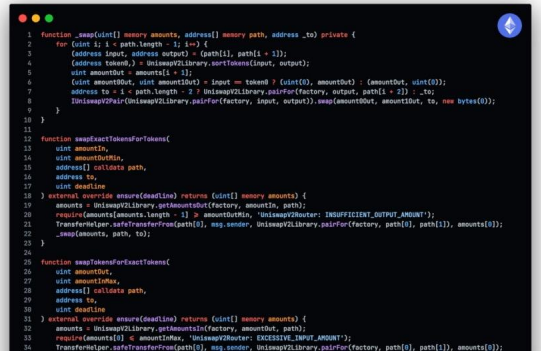
RADIX



```
pub fn swap(&mut self, input_tokens: Bucket) -> Bucket {
    // Getting the vault corresponding to the input tokens and the vault corresponding to the output tokens
    // based on what the input is.
    let (input_tokens_vault, output_tokens_vault): (&mut Vault, &mut Vault) =
        if input_tokens.resource_address() == self.a_pool.resource_address() {
            (&mut self.a_pool, &mut self.b_pool)
        } else if input_tokens.resource_address() == self.b_pool.resource_address() {
            (&mut self.b_pool, &mut self.a_pool)
        } else {
            panic!("The given input tokens do not belong to this liquidity pool")
        };

    // Calculate the output amount of tokens based on the input amount and the pool fees
    let output_amount: Decimal = (input_tokens.amount()
        * (dec!("1") - self.fee)
        * output_tokens_vault.amount())
        / (input_tokens_vault.amount() + input_tokens.amount() * (dec!("1") - self.fee));

    // Perform the swapping operation
    input_tokens_vault.put(input_tokens);
    output_tokens_vault.take(output_amount)
}
```

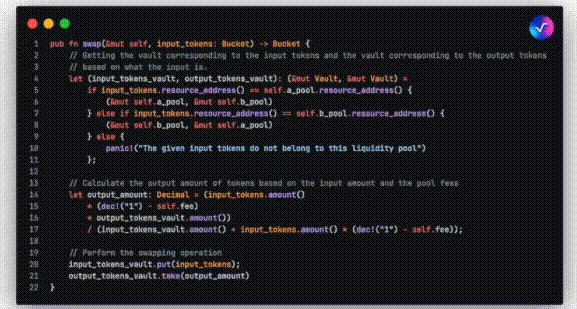


```
function _swap(uint[] memory amounts, address[] memory path, address _to) private {
    for (uint i; i < path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0,) = UniswapV2Library.sortTokens(input, output);
        uint amountOut = amounts[i + 1];
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
        address to = i < path.length - 2 ? UniswapV2Library.pairFor(factory, output, path[i + 2]) : _to;
        IUniswapV2Pair(UniswapV2Library.pairFor(factory, input, output)).swap(amount0Out, amount1Out, to, new bytes(0));
    }
}

function swapExactTokensForTokens(
    uint amountIn,
    uint amountOutMin,
    address[] calldata path,
    address to,
    uint deadline
) external override ensure(deadline) returns (uint[] memory amounts) {
    amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
    require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT');
    TransferHelper.safeTransferFrom(path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0], path[1]), amounts[0]);
    _swap(amounts, path, to);
}

function swapTokensForExactTokens(
    uint amountOut,
    uint amountInMax,
    address[] calldata path,
    address to,
    uint deadline
) external override ensure(deadline) returns (uint[] memory amounts) {
    amounts = UniswapV2Library.getAmountsIn(factory, amountOut, path);
    require(amounts[0] <= amountInMax, 'UniswapV2Router: EXCESSIVE_INPUT_AMOUNT');
    TransferHelper.safeTransferFrom(path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0], path[1]), amounts[0]);
```

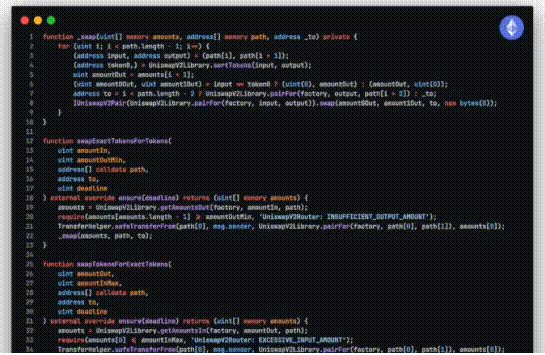
RADIX



```
pub fn swap(&mut self, input_tokens: Bucket) -> Bucket {
    // Getting the vault corresponding to the input tokens and the vault corresponding to the output tokens
    // based on what the input is.
    let (input_tokens_vault, output_tokens_vault): (&mut Vault, &mut Vault) =
        if input_tokens.resource_address() == self.a_pool.resource_address() {
            (&mut self.a_pool, &mut self.b_pool)
        } else if input_tokens.resource_address() == self.b_pool.resource_address() {
            (&mut self.b_pool, &mut self.a_pool)
        } else {
            panic!("The given input tokens do not belong to this liquidity pool")
        };

    // Calculate the output amount of tokens based on the input amount and the pool fees
    let output_amount: Decimal = (input_tokens.amount()
        * (dec!("1") - self.fee)
        * output_tokens_vault.amount())
        / (input_tokens_vault.amount() + input_tokens.amount() * (dec!("1") - self.fee));

    // Perform the swapping operation
    input_tokens_vault.put(input_tokens);
    output_tokens_vault.take(output_amount)
}
```

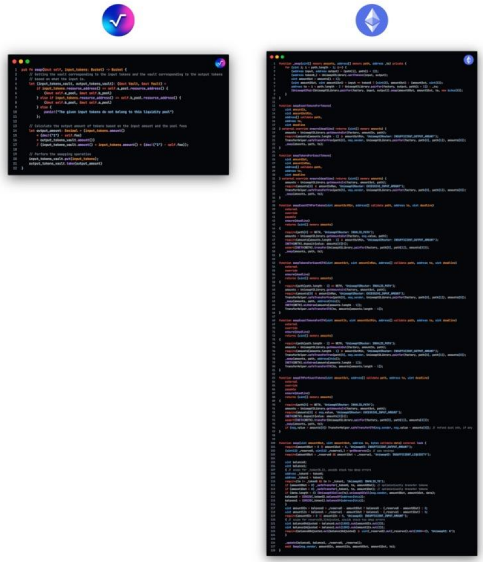


```
function _swap(uint[] memory amounts, address[] memory path, address _to) private {
    for (uint i; i < path.length - 1; i++) {
        (address input, address output) = (path[i], path[i + 1]);
        (address token0,) = UniswapV2Library.sortTokens(input, output);
        uint amountOut = amounts[i + 1];
        (uint amount0Out, uint amount1Out) = input == token0 ? (uint(0), amountOut) : (amountOut, uint(0));
        address to = i < path.length - 2 ? UniswapV2Library.pairFor(factory, output, path[i + 2]) : _to;
        IUniswapV2Pair(UniswapV2Library.pairFor(factory, input, output)).swap(amount0Out, amount1Out, to, new bytes(0));
    }
}

function swapExactTokensForTokens(
    uint amountIn,
    uint amountOutMin,
    address[] calldata path,
    address to,
    uint deadline
) external override ensure(deadline) returns (uint[] memory amounts) {
    amounts = UniswapV2Library.getAmountsOut(factory, amountIn, path);
    require(amounts[amounts.length - 1] >= amountOutMin, 'UniswapV2Router: INSUFFICIENT_OUTPUT_AMOUNT');
    TransferHelper.safeTransferFrom(path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0], path[1]), amounts[0]);
    _swap(amounts, path, to);
}

function swapTokensForExactTokens(
    uint amountOut,
    uint amountInMax,
    address[] calldata path,
    address to,
    uint deadline
) external override ensure(deadline) returns (uint[] memory amounts) {
    amounts = UniswapV2Library.getAmountsIn(factory, amountOut, path);
    require(amounts[0] <= amountInMax, 'UniswapV2Router: EXCESSIVE_INPUT_AMOUNT');
    TransferHelper.safeTransferFrom(path[0], msg.sender, UniswapV2Library.pairFor(factory, path[0], path[1]), amounts[0]);
```

RADIX



RADIX

To Get Started

Visit https://developers.radixdlt.com/

SCAN ME

RADIX