

Sui Campus Workshop

in 千葉工業大学 Session #3

自己紹介



2021年にCryptoGamesに入社し、ブロックチェーンエンジニアとして活動。AstarGames(現MOCHIRON社)、SuperTeam Japanを経て、現在は株式会社シーエーシーのブロックチェーン推進グループに所属。

シーエーシーは2017年からのブロックチェーン事業を開始し、**Corda**を利用したサービスを手がけ、**Avalanche, Solana, Suiの3基盤の技術選定**を行っています。

CAC、エンタープライズ向けブロックチェーン「Corda」の公式開発パートナー認定制度のプレミアパートナー認定を取得

HOME / ニュース / CAC、エンタープライズ向けブロックチェーン「Corda」の公式開発パートナー認定制度のプレミアパートナー認定を取得

2024/09/24

株式会社シーエーシー

社会や産業のデジタルイノベーションに取り組む株式会社シーエーシー（本社：東京都中央区、代表取締役社長：西森良太、以下CAC）は、エンタープライズ向けブロックチェーン「Corda」を活用したシステム開発力と実績が評価され、SBI R3 Japan株式会社（本社：東京都港区、代表取締役：藤本守、以下SBI R3 Japan）から公式開発パートナー認定において、唯一のプレミアパートナー認定を取得したことをお知らせいたします。

■CACのブロックチェーン分野への取り組み

CACは、2016年からブロックチェーン技術への取り組みを開始し、技術者育成とノウハウの蓄積に努めつつ、エンタープライズ・ブロックチェーンシステムの本番向け開発と実運用の実績を重ねてきました。「Corda」については、「Corda」初の国内商用システムであるSBIグループの外国為替コンファメーションシステム「BCPostTrade」の開発に協力したほか、自社開発のエンゲージメントシステム「KOUKA」の基盤にも採用するなど、金融・非金融問わず様々なユースケースにおいて開発・運用してきました。

■「Corda」公式開発パートナー認定取得の背景

CACには、SBI R3 Japan認定資格保有のエンジニアやトレーニングを受講したエンジニアが多数在籍し、社内外で「Corda」を活用したシステム開発に積極的に取り組んできました。こうした実績が評価され、CACは「プレミアパートナー」「アドバンストパートナー」「パートナー」が設定された「Corda」公式開発パートナー認定において、「Corda」開発実績がある複数のベンダーの中で、唯一のプレミアパートナーに認定されました。

CACでは今後も様々な分野でブロックチェーン技術による社会課題の解決やビジネス革新への取り組みを進めてまいります。

https://www.cac.co.jp/news/topics_240924/

唯一の「Corda」プレミアパートナー認定取得

PR TIMES

<https://prtimes.jp/main/html/rd/p/000000101.000024483.html>

株式会社シーエーシー 2024年09/24 10:00

CAC、エンタープライズ向けブロックチェーン「Corda」の公式開発パートナー認定制度のプレミアパートナー認定を取得

社会や産業のデジタルイノベーションに取り組む株式会社シーエーシー（本社：東京都中央区、代表取締役社長：西森良太、以下CAC）は、エンタープライズ向けブロックチェーン「Corda」を活用したシステム開発力と実績が評価され、SBI R3 Japan株式会社（本社：東京都港区、代表取締役：藤本守、以下SBI R3 Japan）から公式開発パートナー認定において、唯一のプレミアパートナー認定を取得したことをお知らせいたします。

■CACのブロックチェーン分野への取り組み

CACは、2016年からブロックチェーン技術への取り組みを開始し、技術者育成とノウハウの蓄積に努めつつ、エンタープライズ・ブロックチェーンシステムの本番向け開発と実運用の実績を重ねてきました。「Corda」については、「Corda」初の国内商用システムであるSBIグループの外国為替コンファメーションシステム「BCPostTrade」の開発に協力したほか、自社開発のエンゲージメントシステム「KOUKA」の基盤にも採用するなど、金融・非金融問わず様々なユースケースにおいて開発・運用してきました。

■「Corda」公式開発パートナー認定取得の背景

CACには、SBI R3 Japan認定資格保有のエンジニアやトレーニングを受講したエンジニアが多数在籍し、社内外で「Corda」を活用したシステム開発に積極的に取り組んできました。こうした実績が評価され、CACは「プレミアパートナー」「アドバンストパートナー」「パートナー」が設定された「Corda」公式開発パートナー認定において、「Corda」開発実績がある複数のベンダーの中で、唯一のプレミアパートナーに認定されました。

CACでは今後も様々な分野でブロックチェーン技術による社会課題の解決やビジネス革新への取り組みを進めてまいります。

■Corda要員・開発実績

- ・Corda 5トレーニング受講修了者: 40名以上
- ・Corda Certifications (R3認定資格取得者数)
 - Business Professional: 10名以上
 - Corda Developer: 10名以上
- ・Cordaシステム開発実績: 10件以上

プレミアパートナー
業界をリードする企業は技術力と実績を認め、トップクラスのパートナー企業です。

アドバンストパートナー
高度な技術力と充実した実績を持つパートナー企業です。

パートナー
Cordaの開発実績を持つパートナー企業として、お客様のニーズにお応えします。

「Corda」認定取得パートナー認定ページ
<https://sbi3.jp/cac.co.jp/corda/corda-partner/>

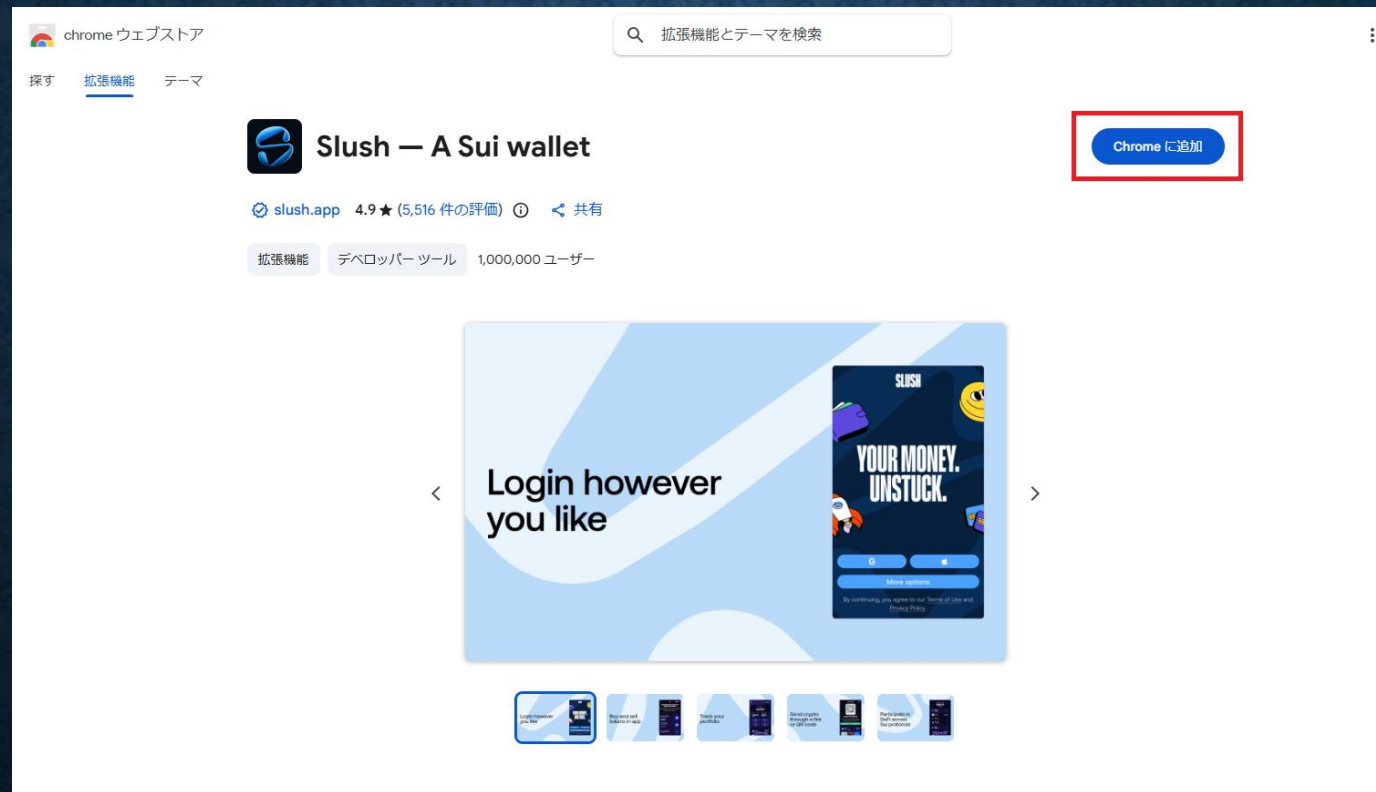
■SBI R3 Japan 代表取締役 藤本守氏のコメント

SBI R3 Japanは、2019年の設立以来、社会コストの削減に貢献するエンジンシステムの構築に取り組んでまいりました。CAC様には、長年にわたる「Corda」を活用したブロックチェーンシステムの開発と実運用にたいへんご協力いただき、その信頼を大変高く感じております。高度な技術力と深い専門知識を有する「Corda」の開発者・開発する企業は国内で非常に少なく、このプレミアムパートナー認定をぜひ式ごとくことになりました。

事前準備

1 Suiウォレットの追加

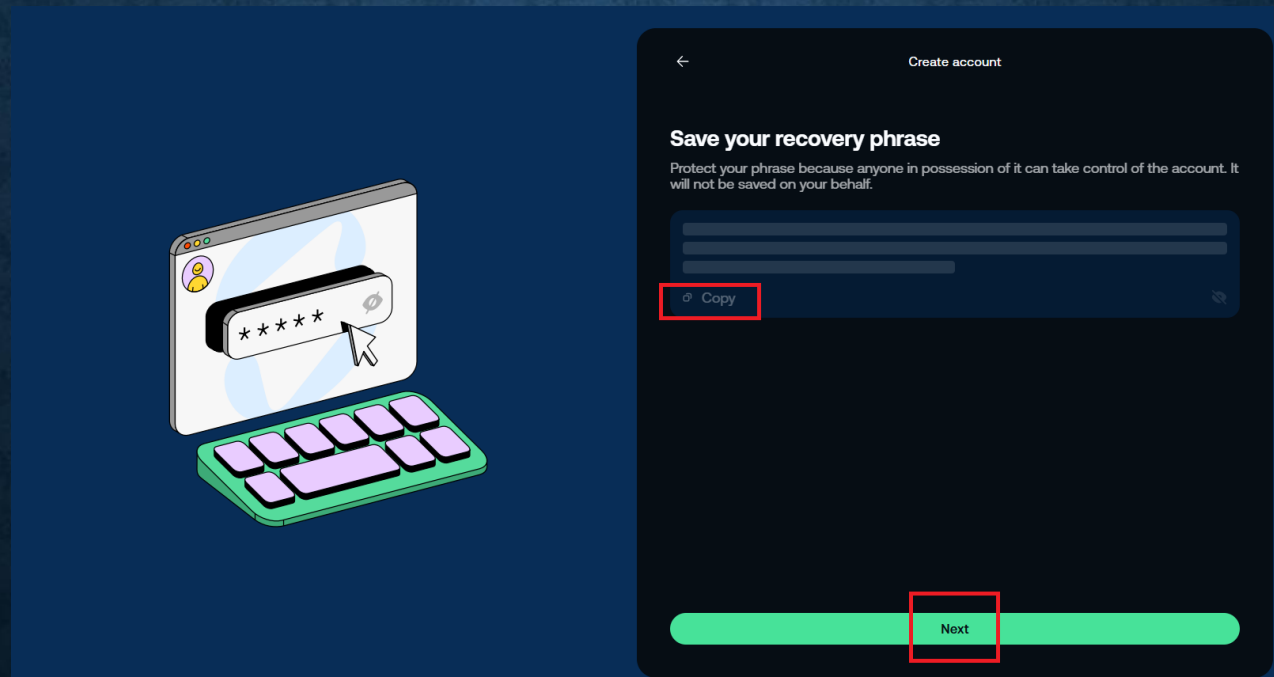
Chromeに追加します。



<https://chromewebstore.google.com/detail/slush-%E2%80%94-a-sui-wallet/opcgpfmipidbgpenhmajoajpbobppdil?hl=ja&pli=1>

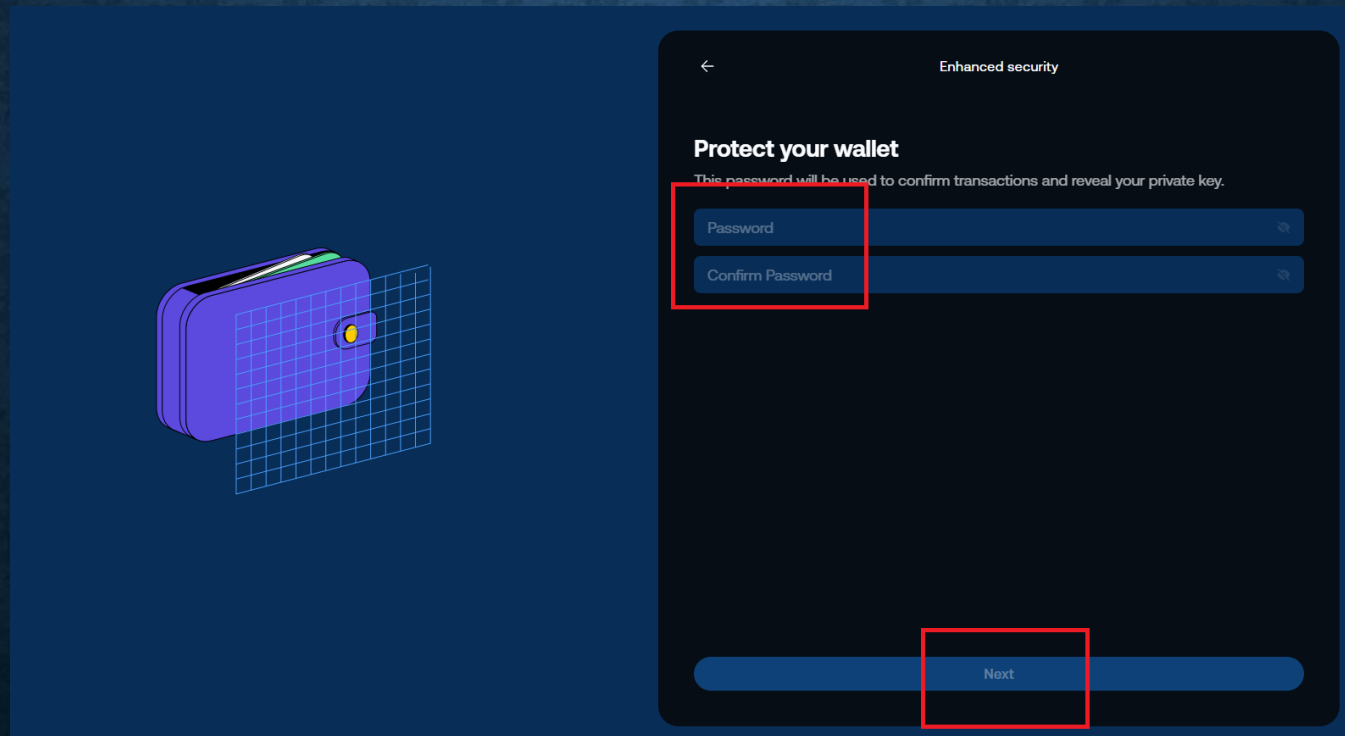
1 Suiウォレットの追加

私や他のメンター含め、絶対に他の人には見せず、厳重に保管してください。



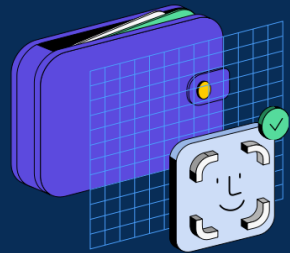
1 Suiウォレットの追加

パスワードを設定して、次へ



1 Suiウォレットの追加

アンロックの時間を設定して次へ



Secure wallet

Protect your wallet with additional security:
Use a password to secure your wallet.

Unlock your wallet ☒

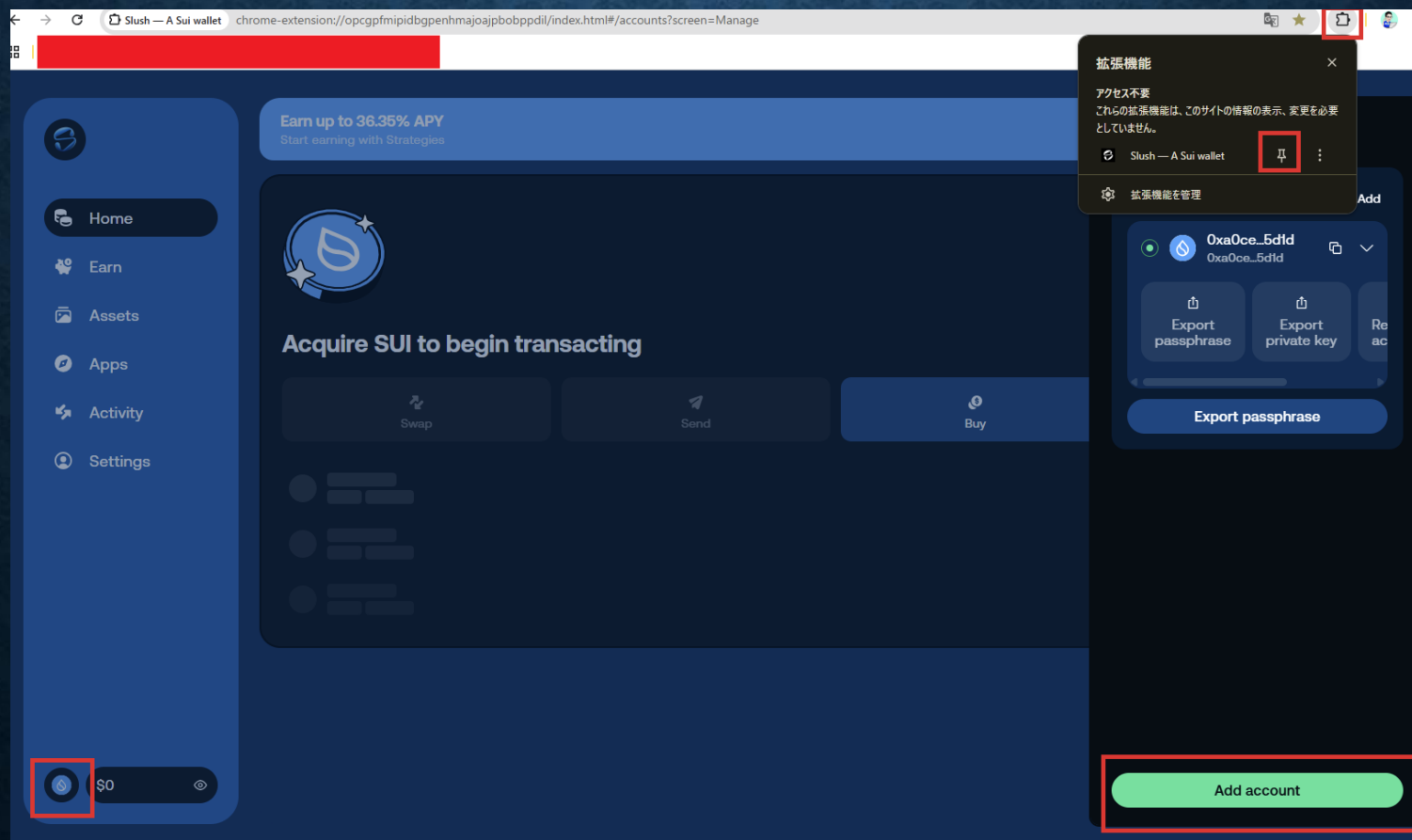
15 minutes ☐
30 minutes ☐
45 minutes ☐
1 hour ☒
4 hours ☐
1 day ☐

Confirm transactions ☐

Next

2 Suiアカウントの作成

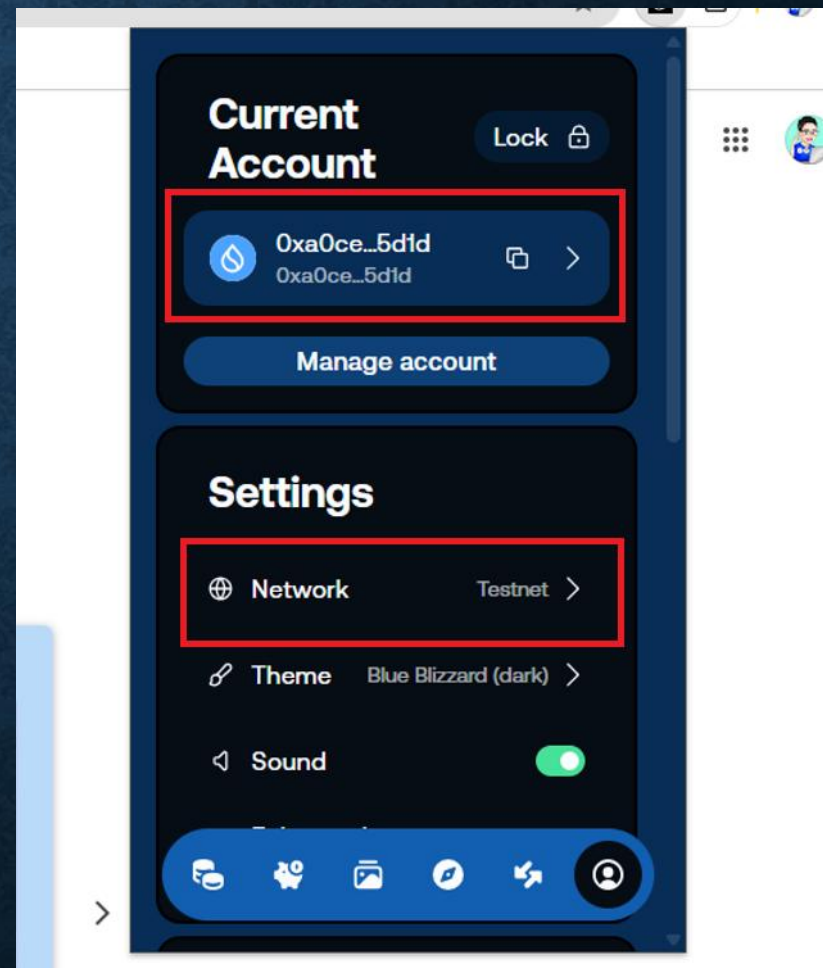
アカウントを作しましょう。
また、ピン止めをしておくと便利です。



2 Suiアカウントの作成

アカウントができました。

Networkはテストネットにしましょう。



2 Suiアカウントの作成

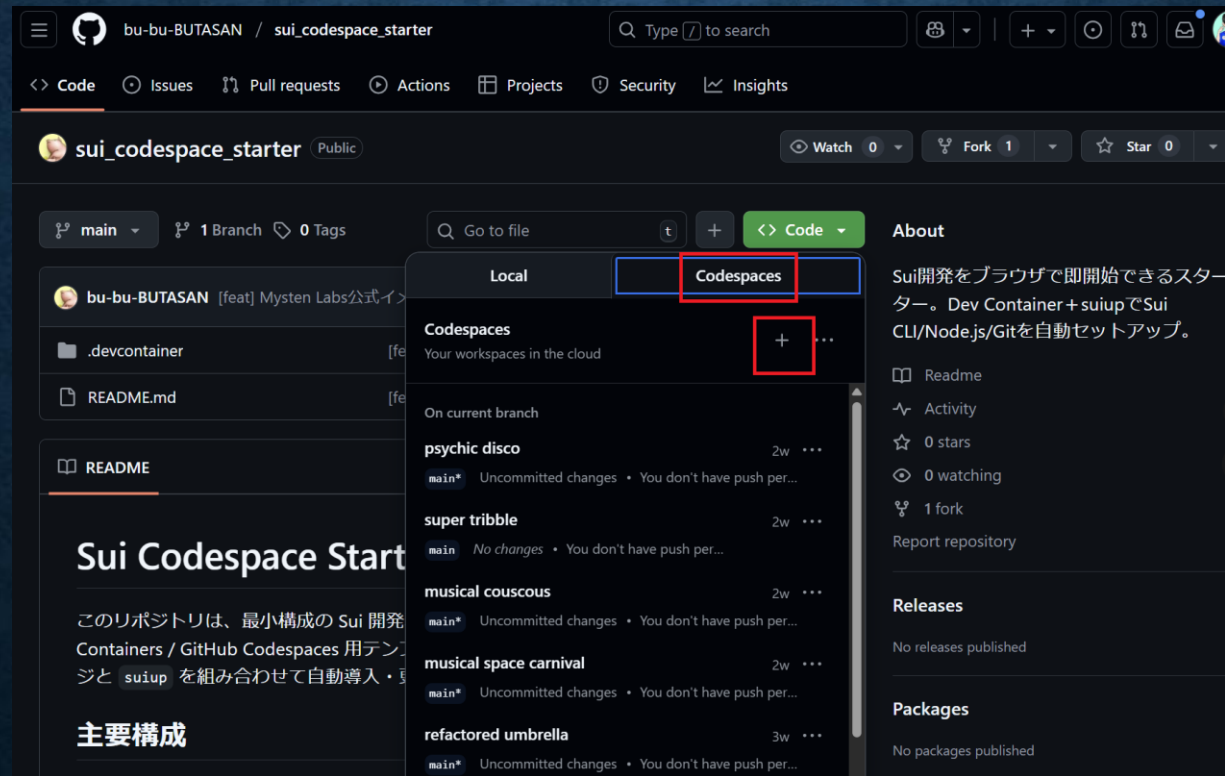
今回初めて作った方はアドレスリストに入れてください。
テストトークンを送付いたします。

	A	B	C	D	E
1	No	名前 (ニックネーム可)	アドレス	チェック	
2	1	ユウキ	0x0863a92345a2f9f9fc88e659cbcc983b2512425a2f2ffe02edbb5da4e6915cb4	✓	
3	2				
4	3				
5	4				
6	5				
7	6				
8	7				
9	8				
10	9				
11	10				
12	11				
13	12				
14	13				
15	14				
16	15				
17	16				
18	17				
19	18				
20	19				
21	20				
22	21				

開発体験

1 Github Codespacesからの立ち上げ

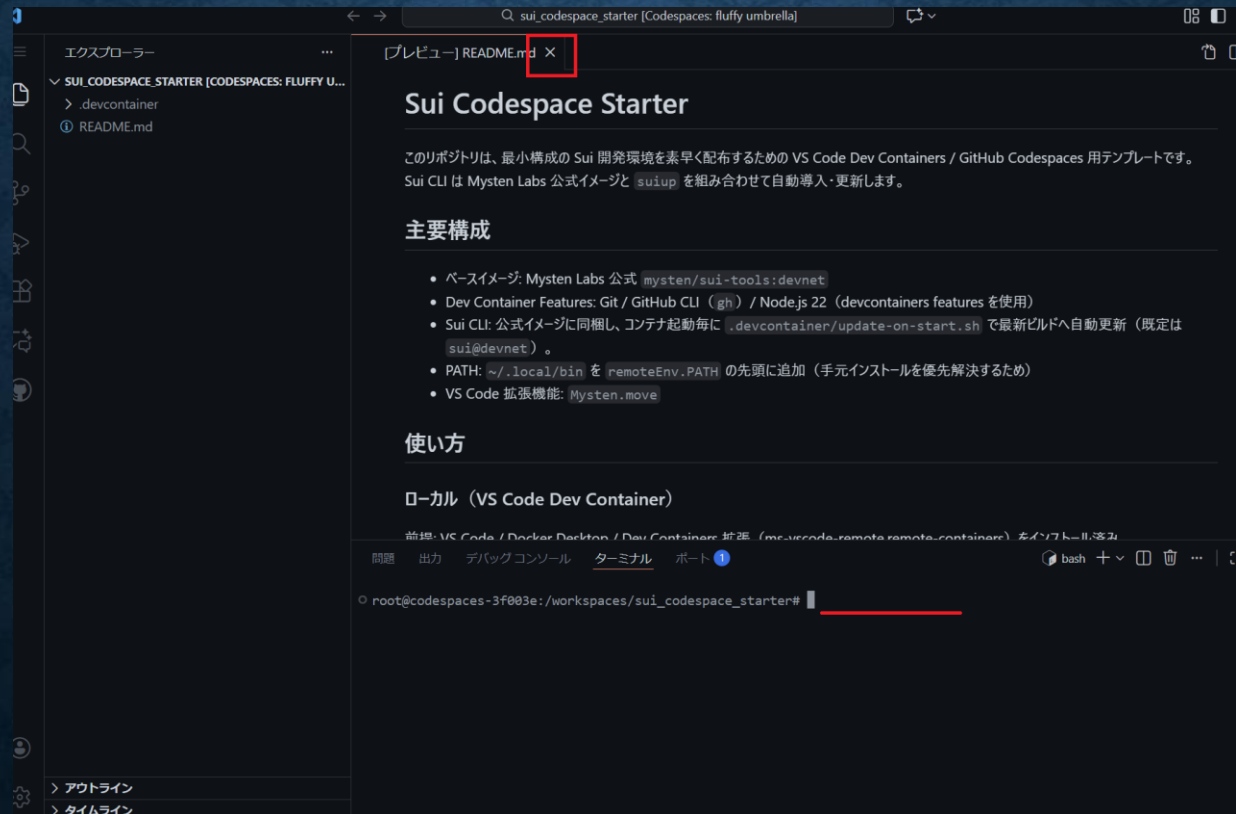
以下のようにして、Githubから立ち上げます。
(Githubアカウントが必要です。)



https://github.com/bu-bu-BUTASAN/sui_codespace_starter

1 Github Codespacesからの立ち上げ

立ち上げることで、コードを入力できるようになります。
「README.md」は使用しないため消しましょう。



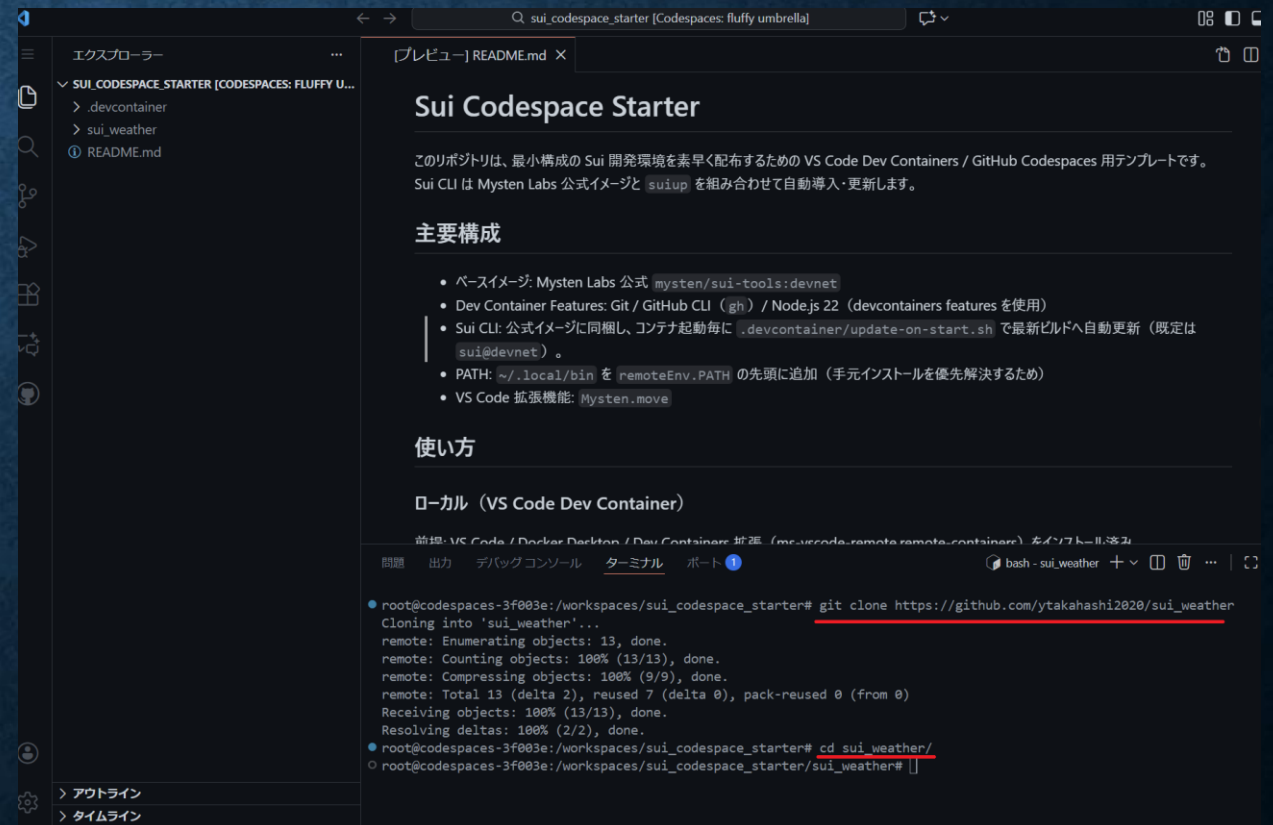
2 Git Cloneの実行

// Git clone

git clone https://github.com/ytakahashi2020/sui_weather

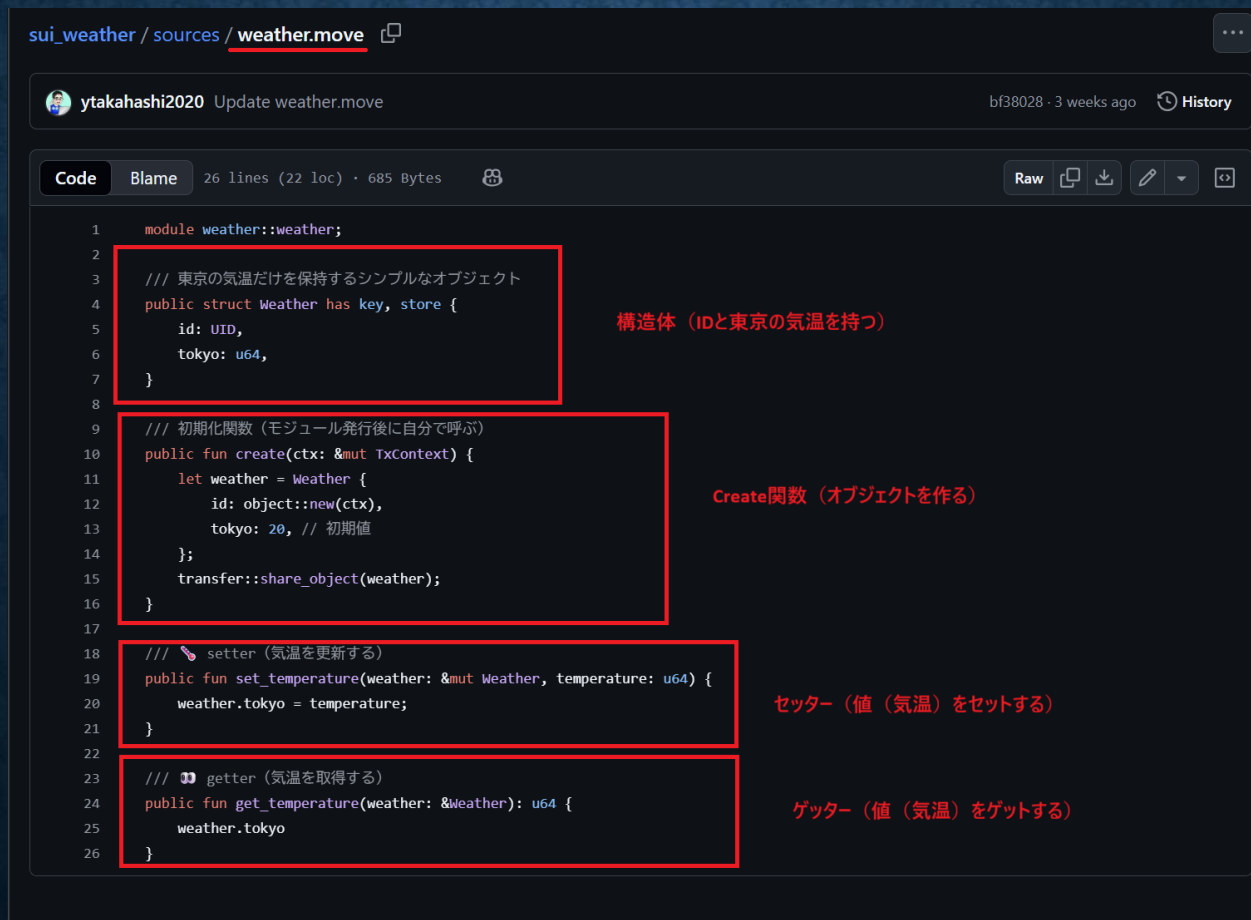
// ディレクトリの移動

cd sui_weather/



2 Git Cloneの実行

大まかな構成を確認しましょう。



```
sui_weather / sources / weather.move
ytakahashi2020 Update weather.move bf38028 · 3 weeks ago History

Code Blame 26 lines (22 loc) · 685 Bytes

1 module weather::weather;
2
3 /// 東京の気温だけを保持するシンプルなオブジェクト
4 public struct Weather has key, store {
5     id: uID,
6     tokyo: u64,
7 }
8
9 /// 初期化関数 (モジュール発行後に自分で呼ぶ)
10 public fun create(ctx: &mut TxContext) {
11     let weather = Weather {
12         id: object::new(ctx),
13         tokyo: 20, // 初期値
14     };
15     transfer::share_object(weather);
16 }
17
18 /// setter (気温を更新する)
19 public fun set_temperature(weather: &mut Weather, temperature: u64) {
20     weather.tokyo = temperature;
21 }
22
23 /// getter (気温を取得する)
24 public fun get_temperature(weather: &Weather): u64 {
25     weather.tokyo
26 }
```

構造体 (IDと東京の気温を持つ)

Create関数 (オブジェクトを作る)

セッター (値 (気温) をセットする)

ゲッター (値 (気温) をゲットする)

3 Suiのウォレットの作成

// キーペアの作成

`sui client -y`

また、アクティブアドレスを表示するコマンドを探しましょう。

```
Resolving deltas: 100% (2/2), done.
• root@codespaces-3f003e:/workspaces/sui_codespace_starter# cd sui_weather/
• root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather# sui client -y
Creating config file ["/root/.sui/sui_config/client.yaml"] with default (devnet) Full node server and ed25519 keypair
.Generated new keypair and alias for address with scheme "ed25519" [hardcore-carnelian: 0x11ba0ff5c75d73bbfdc4eb6d8d2fa9137c9c8d01d7ad7ed82fe1cb1]
Secret Recovery Phrase : [vacant orchard fruit beyond usage eternal spike dune talent gossip horn acoustic]
Client for interacting with the Sui network

Usage: sui client [OPTIONS] [COMMAND]

Commands:
  active-address      Default address used for commands when none specified
  active-env          Default environment used for commands when none specified
  addresses            Obtain the Addresses managed by the client
  balance             List the coin balance of an address
  call               Call Move function
  chain-identifier    Query the chain identifier from the rpc endpoint
  dynamic-field       Query a dynamic field by its address
  envs               List all Sui environments
  execute-signed-tx   Execute a Signed Transaction. This is useful when the user prefers to sign elsewhere
                    and use this command to execute
  execute-combined-signed-tx Execute a combined serialized SenderSignedData string
  faucet             Request gas coin from faucet. By default, it will use the active address and the
                    active network
```


3 Suiのウォレットの作成

// アクティブアドレスの表示

sui client active-address

Options:

```
--client.env <ENV>      The Sui environment to use. This must be present in the current config file.
```

```
--json          Return command outputs in json format
```

-y, --yes

```
-h, --help      Print help
```

```
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather# sui client active-address
```

```
[warning] Client/Server api version mismatch, client api version : 1.62.0, server api version : 1.61.2
```

```
0x11ba0ff5c75d73bbfdc4eb60be1dcab6d8d2fa9137c9c8d01d7ad7ed82fe1cb1
```

```
root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
```

```
root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
```

```
root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

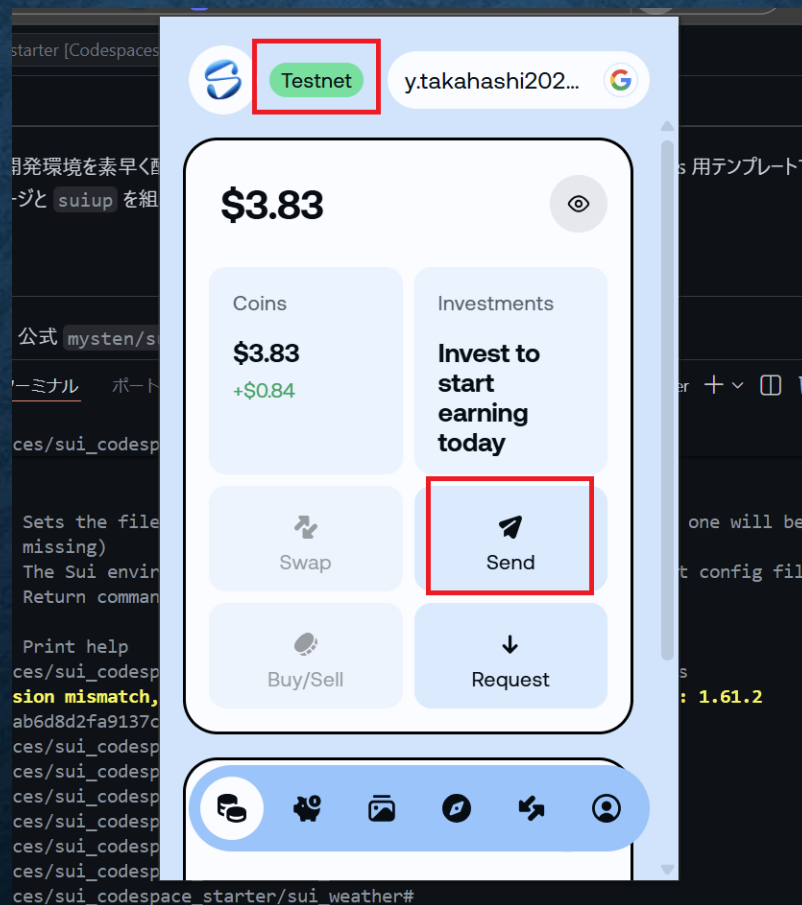
```
root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
root@codespaces-3f003e: /workspaces/sui_codespace_starter/sui_weather#
```

4 ガス代の確認

アクティブアドレスに向けて、少量のガス代を送りましょう。



4 ガス代の確認

// 残高の確認

sui client gas

```

[main@nag: ~]$ curl -s https://api-version.minaexplorer.com/api-version/1.2.0/2.0,server-api-version/1.2.0/2.1
No gas coins are owned by this address

```

```
● root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather# sui client gas
```

```
[warning] Client/Server api version mismatch, client api version : 1.62.0, server api version : 1.61.2
```

gasCoinId	mistBalance (MIST)	suiBalance (SUI)
0x61a2cc113e1a97cd6872485c420345451bf59ef8b86753570bb5c822479813ae	100000000	0.10

```
○ root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
```

```
○ root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
```

```
○ root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
```

```
○ root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

```
○ root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
```

```
○ root@codespaces-3f003e:/workspaces/sui codespace starter/sui weather#
```

5 デプロイの実行

// デプロイの実行

sui client publish

```
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather#
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather# sui client publish
[warning] Client/Server api version mismatch, client api version : 1.62.0, server api version : 1.61.2
[Note]: Dependency sources are no longer verified automatically during publication and upgrade. You can pass the
fy-deps` option if you would like to verify them as part of publication or upgrade.
FETCHING GIT DEPENDENCY https://github.com/MystenLabs/sui.git
Cloning into '/root/.move/https___github_com_MystenLabs_sui_git_c1f1ae650fb9f9248b39a569400b4420820868db'...
remote: Enumerating objects: 34419, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 34419 (delta 0), reused 0 (delta 0), pack-reused 34403 (from 2)
Receiving objects: 100% (34419/34419), 18.34 MiB | 15.53 MiB/s, done.
Resolving deltas: 100% (3502/3502), done.
FETCHING GIT DEPENDENCY https://github.com/MystenLabs/sui.git
Cloning into '/root/.move/https___github_com_MystenLabs_sui_git_f63c9fc78e2171fa174dc43e757ded416c204558'...
remote: Enumerating objects: 34419, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 34419 (delta 0), reused 0 (delta 0), pack-reused 34403 (from 2)
Receiving objects: 100% (34419/34419), 18.34 MiB | 12.01 MiB/s, done.
Resolving deltas: 100% (3502/3502), done.
INCLUDING DEPENDENCY Bridge
INCLUDING DEPENDENCY SuiSystem
INCLUDING DEPENDENCY Sui
INCLUDING DEPENDENCY MoveStdlib
BUILDING weather
Skipping dependency verification
Transaction Digest: 7wpDGyRGQhNgU1mNhm8v7Ly4XG7KMikQwZ3BUVaT3HJr
```


6 パッケージの確認

パッケージIDはこちらです。

```
問題 出力 デバッグ コンソール ターミナル ポート 1 bash - sui_weather + ▾ 🗑️ ... |
root@codespaces-3f003e:/workspaces/sui_codespace_starter/sui_weather# sui client publish
Digest: 3TcgLhcfN3swCNuHu6fseFvxjejpозodEuLjTnSyzUEn

Mutated Objects:

ObjectID: 0x61a2cc113e1a97cd6872485c420345451bf59ef8b86753570bb5c822479813ae
Sender: 0x11ba0ff5c75d73bbfdc4eb60be1dcab6d8d2fa9137c9c8d01d7ad7ed82fe1cb1
Owner: Account Address ( 0x11ba0ff5c75d73bbfdc4eb60be1dcab6d8d2fa9137c9c8d01d7ad7ed82fe1cb1 )
ObjectType: 0x2::coin::Coin<0x2::sui::SUI>
Version: 649439181
Digest: 8TEhw7Xc5B4J853bm8ttDzBxGD2o811fL1QpRksqqsfJ

Published Objects:

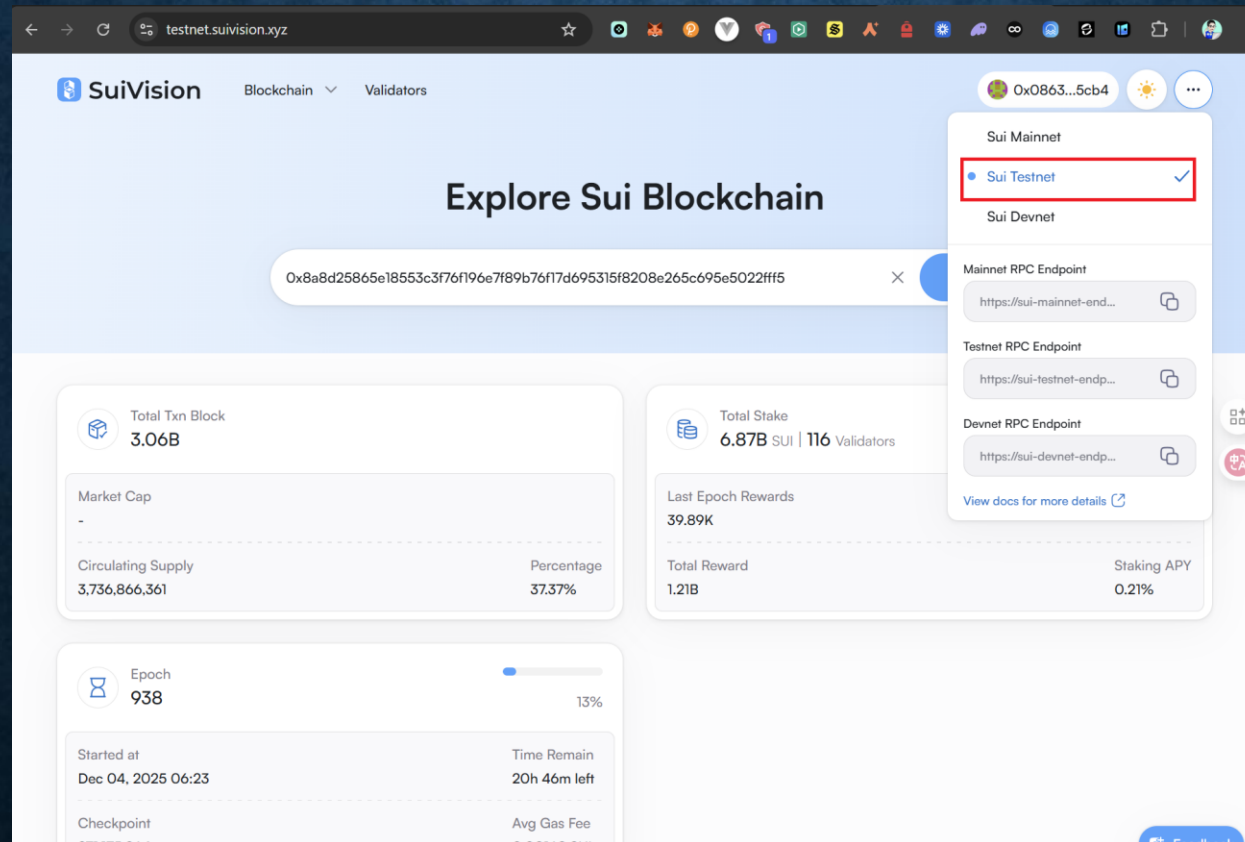
PackageID: 0x98a040f176d36878c50bbb6a7c482e7554c93fe03b5c37c205b623461f992e96
Version: 1
Digest: GAKw7i4mp7Dk3vCMkb9NyXvfxyfNFkxAamn1qLrXV6Mj
Modules: weather

Balance Changes

Owner: Account Address ( 0x11ba0ff5c75d73bbfdc4eb60be1dcab6d8d2fa9137c9c8d01d7ad7ed82fe1cb1 )
CoinType: 0x2::sui::SUI
Amount: -7568680
```

6 パッケージの確認

エクスプローラ(Sui Vision)でも確認してみましょう。



<https://testnet.suivision.xyz/>

7 関数の実行

「Create」関数からオブジェクトを作ってみましょう。

The screenshot shows a web interface for executing a function. At the top, there's a header with metadata: Publish Time (Dec 04, 2025 00:36:00 +UTC), Last Transaction ID (5iQ26d1mji1CW1PrXqCRRMB9oP5wbUNPh1pHbmPNXJBA), Version (1), and Owner (Immutable). Below this, there are tabs for Transaction Blocks, Code (selected), and Statistics. The main area is divided into two sections: a code editor on the left and an execution panel on the right. The code editor shows a Rust-like code snippet for a 'weather' module, including a struct definition and three functions: 'create', 'get_temperature', and 'set_temperature'. The execution panel on the right lists the functions 'create', 'get_temperature', and 'set_temperature'. The 'create' function is selected, and the 'Execute' button is highlighted with a red box.

```
1 module 0x8a8d25865e18553c3f76f196e7f89b76f17d695315f8208e265c695e5022fff5::weather {  
2     struct Weather has store, key {  
3         id: 0x2::object::UID,  
4         tokyo: u64,  
5     }  
6  
7     public fun create(arg0: &mut 0x2::tx_context::TxContext) {  
8         let v0 = Weather{  
9             id : 0x2::object::new(arg0),  
10            tokyo : 20,  
11        };  
12        0x2::transfer::share_object<Weather>(v0);  
13    }  
14  
15    public fun get_temperature(arg0: &Weather) : u64 {  
16        arg0.tokyo  
17    }  
18  
19    public fun set_temperature(arg0: &mut Weather, arg1: u64) {  
20        arg0.tokyo = arg1;  
21    }  
22  
23    // decompiled from Move bytecode v6  
24 }  
25
```

Execute

7 関数の実行

処理(トランザクション)が実施されていることが確認できます。

The screenshot displays the Sui Package Explorer interface. At the top, the 'Package' section shows the address `0x8a8d...fff5`. Below this, the 'Details' tab is active, showing the following information:

- Publisher: `0x3b15...418a`
- Publish Time: Dec 04, 2025 00:36:00 +UTC
- Last Transaction ID: `5iQ26d1mj11CW1PrXqCRRMB9oP5wbUNPh1pHbmPNXJBA`
- Version: 1
- Owner: Immutable

Below the details, there are three tabs: 'Transaction Blocks' (highlighted with a red box), 'Code', and 'Statistics'. The 'Transaction Blocks' tab is active, showing a table of transaction blocks. The table has columns: DIGEST, PAYLOAD, SENDER, TXNS, STATUS, AGE, and GAS. The first row is highlighted with a red box:

DIGEST	PAYLOAD	SENDER	TXNS	STATUS	AGE	GAS
<code>8k2uh6...VNRk</code>	<code>0x8a8d...fff5::weather::create</code>	<code>0x0863...5cb4</code>	1	✓	33 secs ago	0.00237028 SUI

7 関数の実行

新しく、「オブジェクト」が作成されていることが確認できました。

The screenshot displays the SuiVision Blockchain Explorer interface. The main header shows 'SuiVision' with navigation links for 'Blockchain' and 'Validators'. A search bar is located at the top right. The page title is 'Transaction Block' for block ID '8k2uh6...VNRk'. Below the title, there are tabs for 'Overview', 'Changes' (highlighted with a red box), 'Events (0)', 'User Signatures', and 'Raw JSON'. The 'Changes' section is divided into two parts: 'Balance Changes' and 'Object Changes'. The 'Balance Changes' table shows a single entry for account '0x0863...5cb4' with a SUI balance change of -0.00237028. The 'Object Changes' table shows two entries. The first entry, highlighted with a red box, shows a new object '0x3f9c...9cca' created by 'Shared(649439182)' with type '0x8a8d...fff5::weather::Weather'. The second entry shows an object '0x3009...eb2e' mutated by '0x0863...5cb4' with type '0x2::coin::Coin<...>'.

ID	TYPE	COIN	AMOUNT
0x0863...5cb4	Account	SUI	-0.00237028 SUI

OBJECT	OWNER	ACTION	TYPE	VERSION
0x3f9c...9cca	Shared(649439182)	Created	0x8a8d...fff5::weather::Weather	649439182
0x3009...eb2e	0x0863...5cb4	Mutated	0x2::coin::Coin<...>	649439182

7 関数の実行

オブジェクトを確認すると、東京の気温が確認できました。

The screenshot displays the SuiVision web application interface. At the top, there's a navigation bar with 'SuiVision', 'Blockchain', and 'Validators' tabs. A search bar on the right allows searching by Account, Package, Object, Transaction, or SuiNS. The main content area is titled 'Object' and shows details for a specific object ID: 0x3f9c...9cca.

Object Details:

- Owner: Shared(649439182)
- Publisher: 0x0863...5cb4
- Type: 0x8a8d...fff5::weather::Weather
- Version: 649439182
- Transaction Block Digest: 8k2uh6...VNRk

Fields:

```
{
  id: {
    id: "0x3f9ce844daf5b26b0907f1e117e727f375be7329fd87bdb05306abbffe09cca"
  }
  tokyo: "20"
}
```

Dynamic Fields:

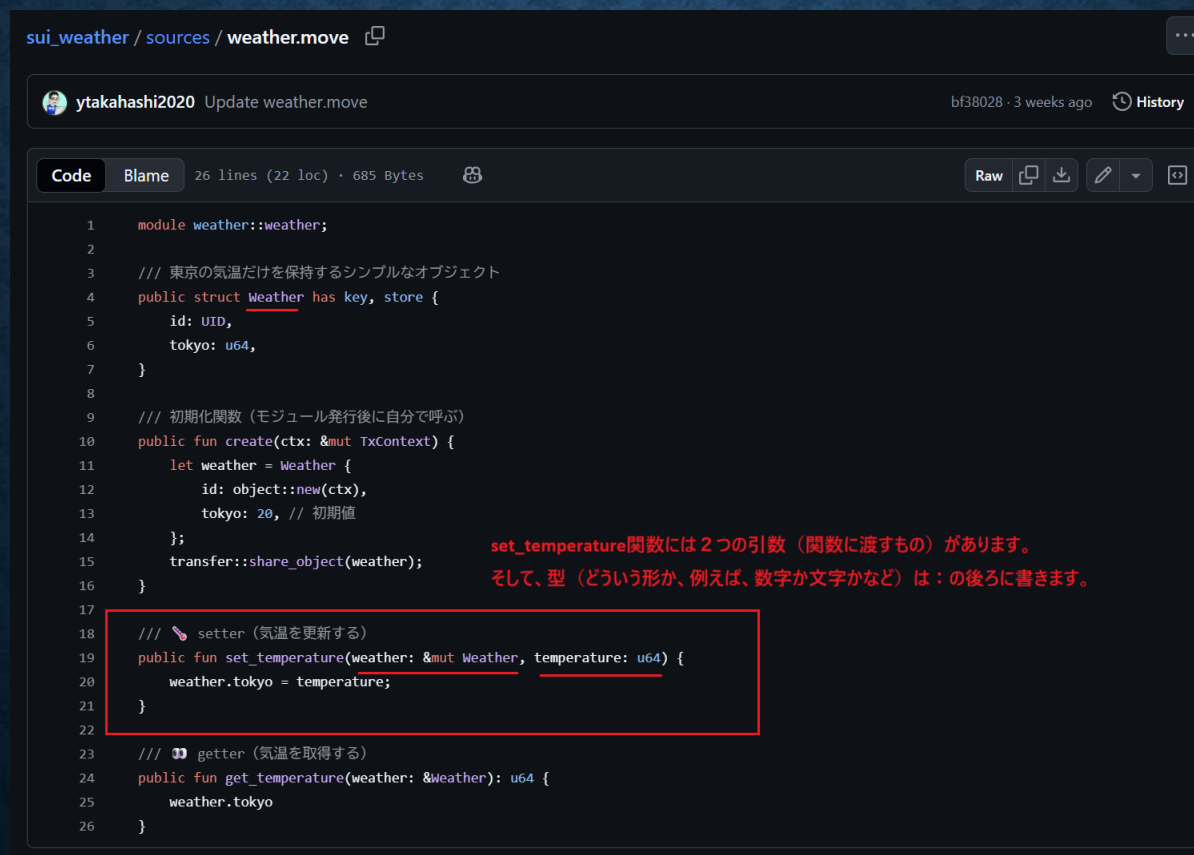
CHILD OBJECT ID	VERSION	OBJECT TYPE	LAST TX ID
No data found			

Transaction Blocks:

Updated Objects | Input Objects

7 関数の実行

別の気温に変えてみましょう。
コードを確認すると、二つの引数が必要なようです。



```
sui_weather / sources / weather.move
ytakahashi2020 Update weather.move bf38028 · 3 weeks ago History

Code Blame 26 lines (22 loc) · 685 Bytes Raw

1  module weather:weather;
2
3  /// 東京の気温だけを保持するシンプルなオブジェクト
4  public struct Weather has key, store {
5      id: UID,
6      tokyo: u64,
7  }
8
9  /// 初期化関数 (モジュール発行後に自分で呼ぶ)
10 public fun create(ctx: &mut TxContext) {
11     let weather = Weather {
12         id: object::new(ctx),
13         tokyo: 20, // 初期値
14     };
15     transfer::share_object(weather);
16 }
17
18 /// 🛠 setter (気温を更新する)
19 public fun set_temperature(weather: &mut Weather, temperature: u64) {
20     weather.tokyo = temperature;
21 }
22
23 /// 📡 getter (気温を取得する)
24 public fun get_temperature(weather: &Weather): u64 {
25     weather.tokyo
26 }
```

set_temperature関数には2つの引数 (関数に渡すもの) があります。
そして、型 (どういう形か、例えば、数字が文字かなど) は : の後ろに書きます。

7 関数の実行

set_temperature関数に二つの引数を渡して実行してみましょう。

The screenshot displays the Move VM interface with the 'weather' module selected. The 'Code' tab is active, showing the decompiled Move code. The 'Execute' panel on the right is open, and the 'set_temperature' function is selected. The arguments for the function are entered in the 'Arg0' and 'Arg1' fields. The 'Arg0' field contains the hexadecimal value '0x3f9ce8446daf5b26b0907f1e117e727f375be7329', and the 'Arg1' field contains the decimal value '30'. A red rectangle highlights these two input fields. The 'Execute' button is visible at the bottom right of the 'Execute' panel.

```
1 module 0x8a8d25865e18553c3f76f196e7f89b76f17d695315f8208e265c695e5022ffff5::weather {
2   struct Weather has store, key {
3     id: 0x2::object::UID,
4     tokyo: u64,
5   }
6
7   public fun create(arg0: &mut 0x2::tx_context::TxContext) {
8     let v0 = Weather{
9       id : 0x2::object::new(arg0),
10      tokyo : 20,
11    };
12    0x2::transfer::share_object<Weather>(v0);
13  }
14
15  public fun get_temperature(arg0: &Weather) : u64 {
16    arg0.tokyo
17  }
18
19  public fun set_temperature(arg0: &mut Weather, arg1: u64) {
20    arg0.tokyo = arg1;
21  }
22
23  // decompiled from Move bytecode v6
24 }
25
```

7 関数の実行

このように気温が変更できました

Object

Search by Account, Package, Object, Transaction, SuiINS / 🔍

0x3f9c...9cca 🔗

Details

Owner:

Shared(649439182)

Publisher:

0x0863...5cb4 🔗

Type:

0x8a8d...fff5::weather::Weather 🔗

Version:

649439184

Transaction Block Digest:

6arW3Z...5zJM 🔗

Fields

```
{
  id: {
    id: "0x3f9ce8446daf5b26b0907f1e117e727f375be7329fd87bdb05306abbffe09cca"
  }
  tokyo: "30"
}
```

Dynamic Fields

CHILD OBJECT ID	VERSION	OBJECT TYPE	LAST TX ID
No data found			

Transaction Blocks

実装体験

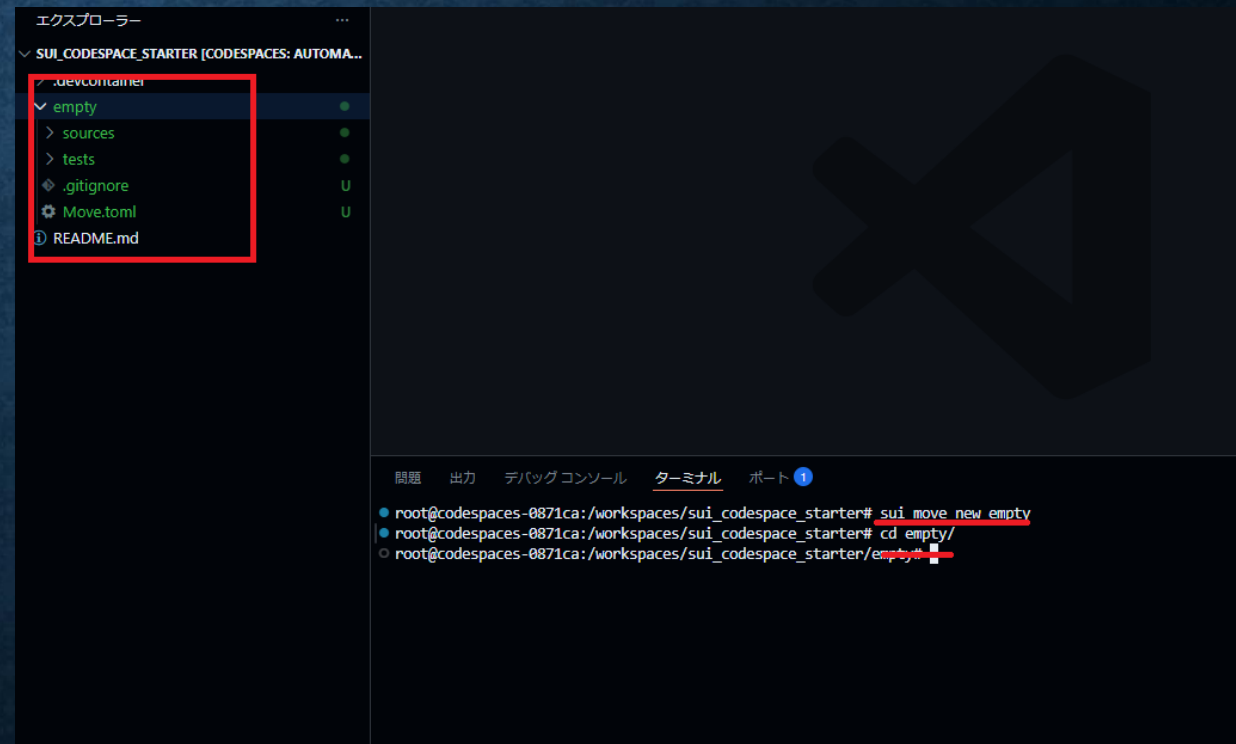
1 パッケージの作成

// パッケージの作成

sui move new <パッケージ名>

// ディレクトリの移動

cd <パッケージ名>



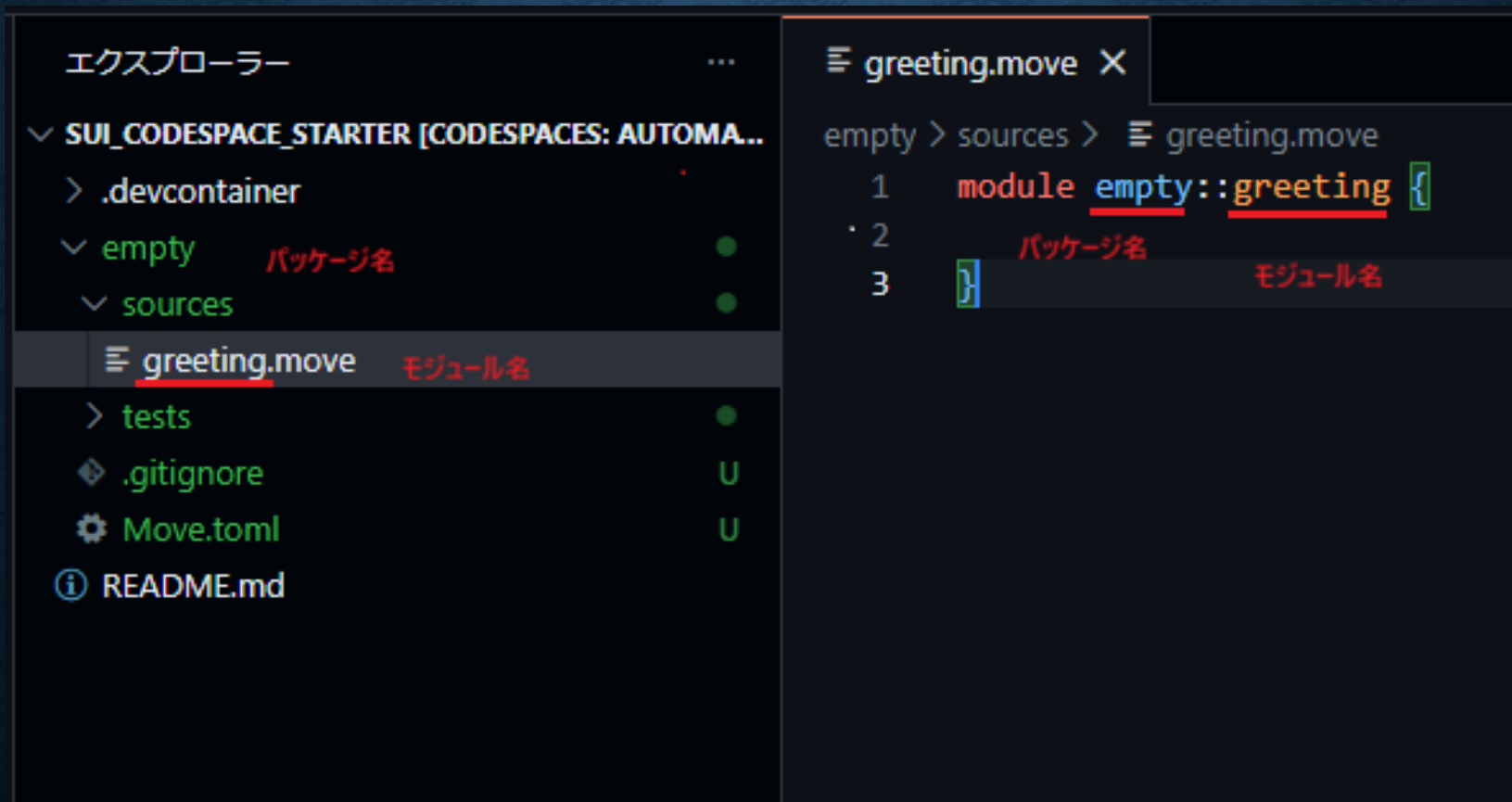
2 パッケージとモジュール

Suiではパッケージの中に複数のモジュールが配置されます。
(とはいえ、多くの場合はモジュールは一つです。)
このモジュールがいわゆるスマートコントラクトです。



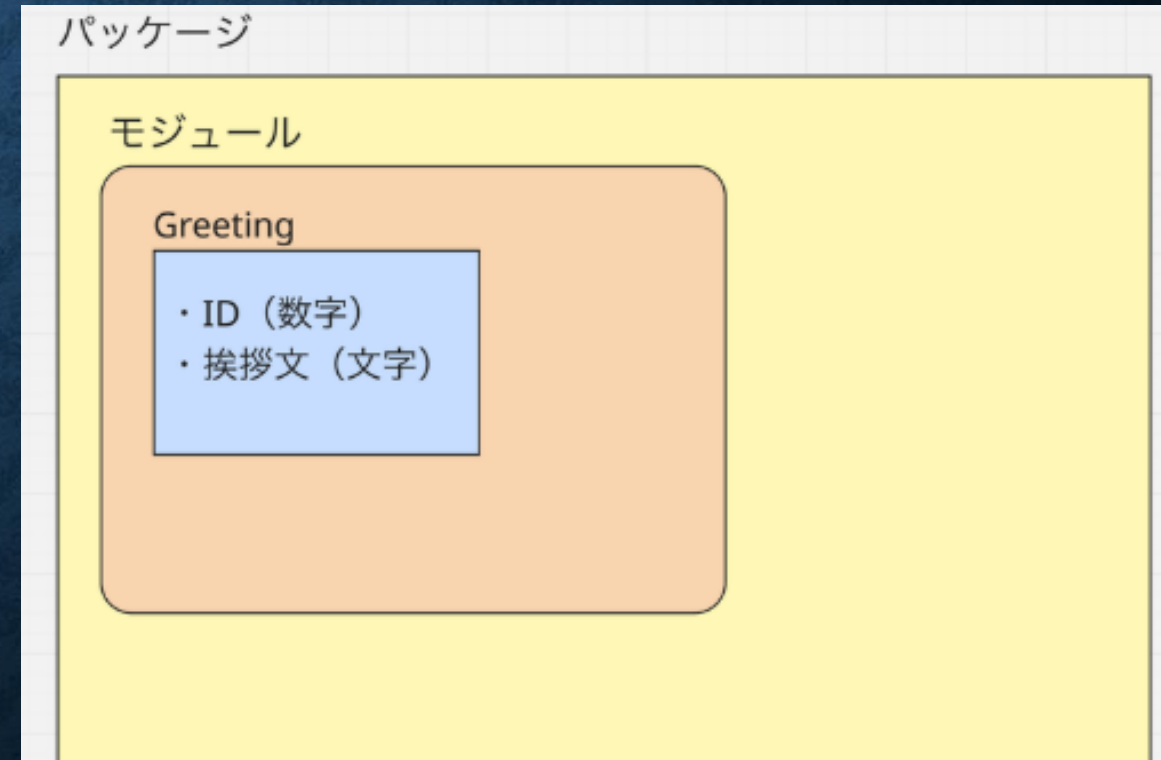
2 パッケージとモジュール

まずはモジュールを作成しましょう。



3 構造体

いくつかの変数(値を入れる入れ物)をまとめたものを構造体と言います。
今回はあいさつ用の構造体を作ろうと思います。



3 構造体

構造体の外側を書いていきましょう。

今回は「Greeting」という名前の構造体を作ります。



```
Home  greeting.move  X
1
2  module empty2::greeting {
3
4      // 構造体
5      public struct Greeting {
6
7      }
8
9  }
10
```

外部から確認可能

構造体のマーク

構造体名

4 Keyの設定

この構造体には4つのアビリティの一つである「key」を設定します。
これでこの構造体が世界で唯一のものとなります。

```
1
2  module empty2::greeting {
3
4      // 構造体
5      public struct Greeting has key {
6          id
7      }
8
9  }
10
```


4 Keyの設定

Suiでは、変数に型を設定します。

その変数に数字が入るのか、文字が入るのかななどを明確にするためです。

idにはUIDという特殊な型を入れます。

```
Home  greeting.move  ×  
1  
2  module empty2::greeting {  
3  
4      // 構造体  
5      public struct Greeting has key {  
6          id: UID  
7      }  
8          UID (Unique ID) という型  
9  }  
10
```

5 関数の作成

特定の処理のことを関数と言います。

ここでは、新しく構造体用のオブジェクト(もの)を作るための関数を作ります。



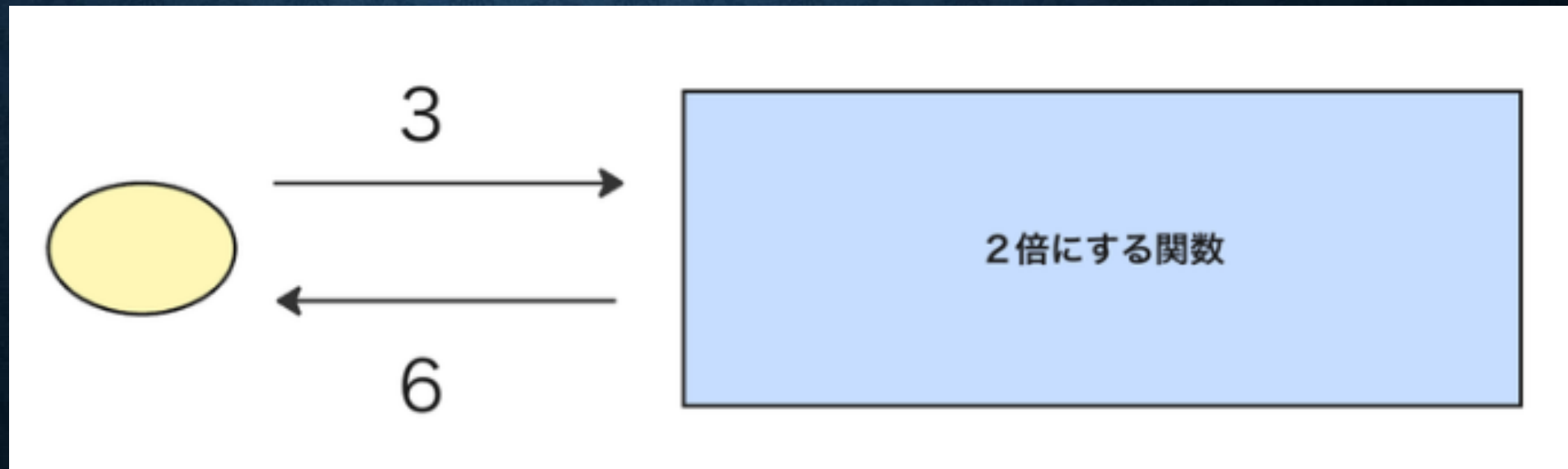
```
1
2 module empty2::greeting {
3
4     // 構造体
5     public struct Greeting has key {
6         id: UID
7     }
8
9     // 関数
10    public fun new() {
11        // 関数名
12    }
13
14 }
15
```

外部から
アクセス可能

関数マーク

5 関数の作成

関数に渡す値のことを引数と言います。
例えば、下の場合、3が引数です。



5 関数の作成

new関数には「TxContext」という型のctxという引数を渡します。

ctxはトランザクションを渡すときの変数のセットです。

新しくオブジェクトを作るときには「&mut TxContext」、そうでないときには「&TxContext」として渡します。

```
4 // 構造体
5 public struct Greeting has key {
6     id: UID
7 }
8
9 // 関数
10 public fun new(ctx: &mut TxContext) {
11
12 }
13
14 }
15
```

5 関数の作成

参考に、TxContextはこのような型です。

```
module sui::tx_context;

/// Information about the transaction currently being executed.
/// This cannot be constructed by a transaction--it is a privileged object created by
/// the VM and passed in to the entrypoint of the transaction as `&mut TxContext`.

public struct TxContext has drop {
    /// The address of the user that signed the current transaction
    sender: address,
    /// Hash of the current transaction
    tx_hash: vector<u8>,
    /// The current epoch number
    epoch: u64,
    /// Timestamp that the epoch started at
    epoch_timestamp_ms: u64,
    /// Counter recording the number of fresh id's created while executing
    /// this transaction. Always 0 at the start of a transaction
    ids_created: u64
}
```

これがポイント
初期値は0でIDを作るたびに1増える

<https://move-book.com/programmability/transaction-context>

5 関数の作成

先ほど作成した構造体を作成します。

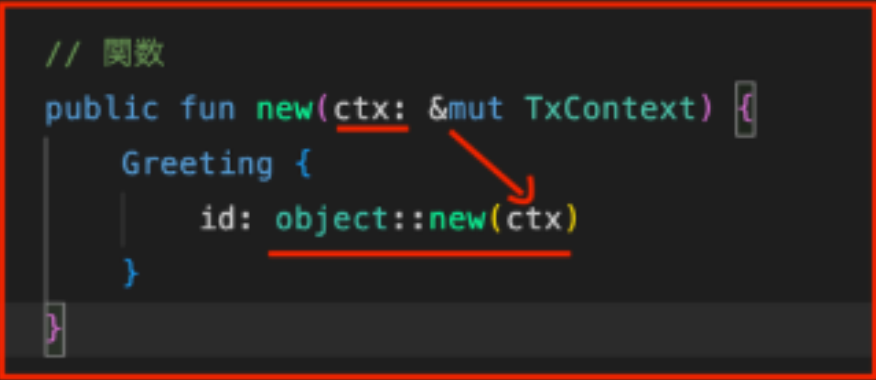
idは適当に設定しましたが、これでは唯一ではなさそうですね。

```
1
2  module empty2::greeting {
3
4      // 構造体
5      public struct Greeting has key {
6          id: UID
7      }
8
9      // 関数
10     public fun new(ctx: &mut TxContext) {
11         Greeting {
12             id: 12345
13         }
14     }
15
16 }
17
```


5 関数の作成

実際には引数のctxを使って、このように作成します。

```
1
2  module empty2::greeting {
3
4      // 構造体
5      public struct Greeting has key {
6          id: UID
7      }
8
9      // 関数
10     public fun new(ctx: &mut TxContext) {
11         Greeting {
12             id: object::new(ctx)
13         }
14     }
15
16 }
17
```



5 関数の作成

作った構造体を変数に格納します。
これによって操作がしやすくなります。

```
1
2 module empty2::greeting {
3
4     // 構造体
5     public struct Greeting has key {
6         id: UID
7     }
8
9     // 関数
10    public fun new(ctx: &mut TxContext) {
11        let new_greeting = Greeting {
12            id: object::new(ctx)
13        };
14    }
15 }
16
17
```

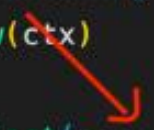
変数名

変数マーク

5 関数の作成

最後に、この構造体の所有者を設定します。
ここでは、共有所有とします。

```
Home  greeting.move X
1
2  module empty2::greeting {
3
4      // 構造体
5      public struct Greeting has key {
6          id: UID
7      }
8
9      // 関数
10     public fun new(ctx: &mut TxContext) {
11         let new_greeting = Greeting {
12             id: object::new(ctx)
13         };
14         transfer::share_object(new_greeting);
15     }
16
17 }
```



5 関数の作成

最後に、この構造体の所有者を設定します。

ここでは、共有所有とします。

```
Home greeting.move X
1
2 module empty2::greeting {
3
4     // 構造体
5     public struct Greeting has key {
6         id: UID
7     }
8
9     // 関数
10    public fun new(ctx: &mut TxContext) {
11        let new_greeting = Greeting {
12            id: object::new(ctx)
13        };
14        transfer::share_object(new_greeting);
15    }
16
17 }
```

6 文字列の設定

文字列を扱うためにuse文を使います。

これを使うことで、文字列であるStringを使う事ができるようになります。

```
Home  greeting.move  X
1
2  module empty2::greeting {
3      use std::string::String;
4
5      // 構造体
6      public struct Greeting has key {
7          id: UID,
8          text: String,
9      }
10
11     // 関数
12     public fun new(ctx: &mut TxContext) {
13         let new_greeting = Greeting {
14             id: object::new(ctx),
15             text: b"this is test".to_string()
16         };
17         transfer::share_object(new_greeting);
18     }
19
20 }
21
```


6 文字列の設定

構造体、関数に文字列を設定します。

b'''という形により、バイト列という形であり、to_string関数で文字列にしています。

```
Home greeting.move X
1
2 module empty2::greeting {
3     use std::string::String;
4
5     // 構造体
6     public struct Greeting has key {
7         id: UID,
8         text: String,
9     }
10
11    // 関数
12    public fun new(ctx: &mut TxContext) {
13        let new_greeting = Greeting {
14            id: object::new(ctx),
15            text: b"this is test".to_string()
16        };
17        transfer::share_object(new_greeting);
18    }
19
20 }
21
```


7 変更を行う関数の作成

構造体の文字列を変える関数を作成します。

引数に構造体(オブジェクト)と変更後の文字列を渡します。

```
9      }
10
11      // 関数
12      public fun new(ctx: &mut TxContext) {
13          let new_greeting = Greeting {
14              id: object::new(ctx),
15              text: b"this is test".to_string()
16          };
17          transfer::share_object(new_greeting);
18      }
19
20      // 更新を行う関数
21      public fun update_text(greeting: &mut Greeting, new_text: String) {
22      }
23
24  }
25
26
```

7 変更を行う関数の作成

引数のnew_textをgreetingの「text」に代入します。
「=」は代入を表します。

```
4
5 // 構造体
6 public struct Greeting has key {
7     id: UID,
8     text: String,
9 }
10
11 // 関数
12 public fun new(ctx: &mut TxContext) {
13     let new_greeting = Greeting {
14         id: object::new(ctx),
15         text: b"this is test".to_string()
16     };
17     transfer::share_object(new_greeting);
18 }
19
20 // 更新を行う関数
21 public fun update_text(greeting: &mut Greeting, new_text: String) {
22     greeting.text = new_text;
23 }
24
25 }
26
```


8 SuiVisionでの実行

SuiVisionで作成しましょう。

new関数で実行するとオブジェクトが作成され、実行できたObjectにIDがついていることも確認しましょう。

The screenshot displays the SuiVision interface, which is used for interacting with the Sui blockchain. It is divided into two main panels.

Left Panel (Code Editor): This panel shows the source code for a module named `greeting`. The code defines a `new` function that creates a new `Greeting` object and an `update_text` function that updates the text of an existing `Greeting` object. The code is written in Move, a programming language used on the Sui blockchain.

```
1 module 0x0222f30f0e08202064310f10a2060a2a7f0a0c109f20a270f247e10a211::greeting {
2   struct Greeting has key {
3     id: <id>::object::ID,
4     text: <id>::string::String,
5   }
6
7   public fun new(arg1: <id>::tx::context::TxContext) {
8     let id = Greeting {
9       id: <id>::object::new(arg1),
10      text: <id>::string::new("hello world"),
11    };
12    <id>::transfer::share_object<Greeting>(id);
13  }
14
15  public fun update_text(arg1: <id>::Greeting, arg2: <id>::string::String) {
16    arg1.text = arg2;
17  }
18
19  // Decompiled from Move Bytecode v8
20 }
21
```

Right Panel (Transaction Block): This panel shows the results of a transaction block. It includes a search bar at the top and tabs for Overview, Changes, Events (0), User Signatures, and Raw JSON. The `Changes` tab is selected, showing a table of balance changes and a table of object changes.

Balance Changes Table:

ID	TYPE	COIN	AMOUNT
0x0000...5cb4	Account	SUI	-0.00242348 SUI

Object Changes Table:

OBJECT	OWNER	ACTION	TYPE	VERSION
0x0427...4253	Shared(02336401)	Created	0x9723...5211::greeting::Greeting	62336401
0x3009...0b2e	0x0000...5cb4	Mutated	0x2::coin::Coin<...>	62336401

9 共有オブジェクトの確認

update_text関数も動くことも確認しましょう。

また、SuiVisionでは実際の文字列を入れられないので0xaaaaなどを入れます。

16進数であることを示しています。

The screenshot displays the SuiVision interface with the 'Code' tab selected. The main area shows the decompiled Move bytecode for a module named 'greeting'. The code defines a struct 'Greeting' with fields 'id' (a UID) and 'text' (a UTF-8 string). It also defines two functions: 'new' which creates a new 'Greeting' object and transfers it to the caller, and 'update_text' which updates the 'text' field of a given 'Greeting' object. The 'update_text' function is currently selected in the right-hand 'Execute' panel. Below the function name, the arguments are shown: 'Arg0' is a long hexadecimal string representing a UID, and 'Arg1' is the hexadecimal string '0x48656c6c6f', which represents the ASCII string 'Hello' in hexadecimal. A red box highlights the 'Execute' button at the bottom right of the interface.

```
1 module 0x9723f398fd65933926d365bf1ce28696af2ef95b55150f358871af347e135211::greeting {
2   struct Greeting has key {
3     id: 0x2::object::UID,
4     text: 0x1::string::String,
5   }
6
7   public fun new(arg0: &mut 0x2::tx_context::TxContext) {
8     let v0 = Greeting{
9       id : 0x2::object::new(arg0),
10      text : 0x1::string::utf8(b"Hello world!"),
11    };
12    0x2::transfer::share_object<Greeting>(v0);
13  }
14
15  public fun update_text(arg0: &mut Greeting, arg1: 0x1::string::String) {
16    arg0.text = arg1;
17  }
18
19  // decompiled from Move bytecode v6
20 }
21
```

補足 実行者への所有者の設定

共有ではなく、自分を所有者にしてみましょう。

`ctx.sender()`が実行者を表します。

```
1 module empty::greeting {
2     use std::string;
3
4     public struct Greeting has key {
5         id: UID,
6         text: string::String,
7     }
8
9     public fun new(ctx: &mut TxContext) {
10         let new_greeting = Greeting {
11             id: object::new(ctx),
12             text: b"Hello world!".to_string()
13         };
14         transfer::public_transfer(new_greeting, ctx.sender());
15     }
16
17     public fun update_text(greeting: &mut Greeting, new_text: string::String) {
18         greeting.text = new_text;
19     }
20 }
```


ワークショツプ

オリジナルのコントラクトを作しましょう。

チームごとにスマートコントラクトを実際に作ってみましょう。

またAIを使い、ここまでで説明していない内容を盛り込むことも大歓迎です。

ワーク時間 30分

発表時間 各3～5分

発表内容

- どのようなコントラクトなのかの発表
- コードの説明
- 現在の課題の説明

サンプル① 大学の公式掲示板

常に最新のお知らせのみが表示される大学の公式掲示板です。
これはそのまま利用できるでしょうか。

```
1 module academy::notice_board {  
2     use std::string;  
3  
4     public struct NoticeBoard has key {  
5         id: UID,  
6         message: string::String,  
7     }  
8  
9     // 共有オブジェクトとしてお知らせ板を1つ発行  
10    public fun new(ctx: &mut TxContext) {  
11        let board = NoticeBoard {  
12            id: object::new(ctx),  
13            message: b"Welcome to Academy Notice Board!".to_string(),  
14        };  
15        transfer::share_object(board);  
16    }  
17  
18    // お知らせ文を更新  
19    public fun update_message(board: &mut NoticeBoard, new_message: string::String) {  
20        board.message = new_message;  
21    }  
22 }  
23
```


サンプル② 大学の公式掲示板(+更新回数)

数字を使う場合は、このように、u64という型を使います。
変化させたい場合は、数式を使う事ができます。

```
1 module academy::notice_board {
2     use std::string;
3
4     public struct NoticeBoard has key {
5         id: UID,
6         message: string::String,
7         update_count: u64,
8     }
9
10    public fun new(ctx: &mut TxContext) {
11        let board = NoticeBoard {
12            id: object::new(ctx),
13            message: b"Welcome to our project!".to_string(),
14            update_count: 0,
15        };
16        transfer::share_object(board);
17    }
18
19    public fun update_message(board: &mut NoticeBoard, new_message: string::String)
20    {
21        board.message = new_message;
22        board.update_count = board.update_count + 1;
23    }
24 }
```


サンプル③ 投票コントラクト

数字を使って、投票を行うためのコントラクトを作ってみました。何か問題
はありますでしょうか。

```
1 module poll::simple_poll {  
2   use std::string;  
3  
4   public struct Poll has key {  
5     id: UID,  
6     question: string::String,  
7     yes_count: u64,  
8     no_count: u64,  
9   }  
10  
11   public fun new(ctx: &mut TxContext) {  
12     let poll = Poll {  
13       id: object::new(ctx),  
14       question: b"Should we adopt this new policy?".to_string(),  
15       yes_count: 0,  
16       no_count: 0,  
17     };  
18     transfer::share_object(poll);  
19   }  
20  
21   public fun vote_yes(poll: &mut Poll) {  
22     poll.yes_count = poll.yes_count + 1;  
23   }  
24  
25   public fun vote_no(poll: &mut Poll) {  
26     poll.no_count = poll.no_count + 1;  
27   }  
28 }  
29
```

参考 追加実装

対策① 管理者だけが処理できる設定にしたい

assert!関数を使ってみると条件をつける事ができます。

```
1 module academy::notice_board {
2     use std::string;
3
4     public struct NoticeBoard has key {
5         id: UID,
6         admin: address,
7         message: string::String,
8     }
9
10    // 作成者 = admin
11    public fun new(ctx: &mut TxContext) {
12        let board = NoticeBoard {
13            id: object::new(ctx),
14            admin: ctx.sender(),
15            message: string::utf8(b"Welcome to our project!"),
16        };
17        transfer::share_object(board);
18    }
19
20    // admin だけ更新可能
21    public fun update_message(
22        board: &mut NoticeBoard,
23        new_message: string::String,
24        ctx: &TxContext,
25    ) {
26        assert!(ctx.sender() == board.admin, 1);
27        board.message = new_message;
28    }
29 }
```

assert! (条件, エラーコード)

対策② 1人一回の処理にしたい

下のように、投票済みリストを作れば良い。

投票内容：「○○○○」

賛成票：40

反対票：30

投票済みリスト

①投票済みリストに登録済みか確認



②確認処理

1) 登録済みの場合：エラー

2) 未登録の場合：登録



③未登録であった場合は投票

対策② 1人一回の処理にしたい

下のように「Table」を使うことで達成する事ができます。

```
1 module poll::simple_poll {
2     use std::string;
3     use sui::table::(Self, Table);
4
5     public struct Poll has key {
6         id: UID,
7         question: string::String,
8         yes_count: u64,
9         no_count: u64,
10        voters: Table<address, bool>, // ここに「投票済みアドレス」を記録
11    }
12
13    public fun new(ctx: &mut TxContext) {
14        let poll = Poll {
15            id: object::new(ctx),
16            question: string::utf8(b"Should we adopt this new policy?"),
17            yes_count: 0,
18            no_count: 0,
19            voters: table::new(ctx),
20        };
21        transfer::share_object(poll);
22    }
23
24    public fun vote_yes(poll: &mut Poll, ctx: &TxContext) {
25        let sender = ctx.sender();
26        assert!(table::contains(&poll.voters, sender), 1);
27        table::add(&mut poll.voters, sender, true);
28        poll.yes_count = poll.yes_count + 1;
29    }
```

table::contains(リスト, チェックしたい値)
→これを否定したい時は先頭に「!」

table::add(リスト, 追加したい値, YesかNo)
→YesかNoは「true」か「false」で表します。

登録済みリスト	内容
Aアドレス	Yes
Bアドレス	Yes
Cアドレス	Yes
Dアドレス	No
Eアドレス	Yes

YesかNoをわざわざ登録する理由はわかりにくいかもしれません。

ただ、これにより、あえてNoにしているDアドレスと、登録がそもそもされていないFアドレスを区別する事ができます。

対策③ 許可された者のみの実行としたい

ホワイトリストを作成することで、実行者を絞る事ができます。
逆に、ブラックリストを作ることで明示的に実行を拒否する事ができます。

```
public fun new(ctx: &mut TxContext) {
    let poll = Poll {
        id: object::new(ctx),
        admin: ctx.sender(),
        question: string::utf8(b"Should we adopt this new policy?"),
        yes_count: 0,
        no_count: 0,
        allow: table::new(ctx),
    };
    transfer::share_object(poll);
}

// 管理者だけが許可アドレスを追加
public fun enroll(poll: &mut Poll, who: address, ctx: &TxContext) {
    assert!(ctx.sender() == poll.admin, E_NOT_ADMIN);
    if (!table::contains(&poll.allow, who)) {
        table::add(&mut poll.allow, who, true);
    }
}

public fun vote_yes(poll: &mut Poll, ctx: &TxContext) {
    let sender = ctx.sender();
    assert!(table::contains(&poll.allow, sender), E_NOT_ALLOWED);
    poll.yes_count = poll.yes_count + 1;
}

public fun vote_no(poll: &mut Poll, ctx: &TxContext) {
```


対策④ 管理者を別の者に設定したい

@をつけることで直接アドレスを指定する事ができます。

```
1 module poll3::simple_poll {
2     use std::string;
3     use sui::table::{Self, Table};
4
5     const E_NOT_ADMIN: u64 = 1;
6     const E_NOT_ALLOWED: u64 = 2;
7     const WHITELIST_MANAGER: address = @0x0863a92345a2f9f9fc88e659cbcc983b2512425a2f2ffe02edbb5da4e6915cb4;
8
9     public struct Poll has key {
10         id: UID,
11         admin: address,           // 管理者 (作成者)
12         question: string::String,
13         yes_count: u64,
14         no_count: u64,
15         allow: Table<address, bool>, // 投票を許可するアドレス集合
16     }
17
18     public fun new(ctx: &mut TxContext) {
19         let poll = Poll {
20             id: object::new(ctx),
21             admin: WHITELIST_MANAGER,
22             question: string::utf8(b"Should we adopt this new policy?"),
23             yes_count: 0,
24             no_count: 0,
25             allow: table::new(ctx),
26         };
27     }
```

対策⑤ 時間を設定したい

時間はタイムスタンプを使用します。

現在時刻はclock::timestamp_ms(clock)を使用します。

```
1 module poll { example_poll {
2   use std::string;
3   use sui::clocks { Self, <Clock>;
4
5   // 2025-12-31 23:59:59.999 JST = 2025-12-31T14:59:59.999Z = 1767193199999 ns
6   const DEADLINE_MS: u64 = 1767193199999;
7
8   public struct Poll has key {
9     id: UID,
10    question: string::String,
11    yes_count: u64,
12    no_count: u64,
13    end_ms: u64, // 期限 (UTCエポック)
14  }
15
16  // 期限満了で合否
17  public fun new_until_2025_12_31_jst(CTX: &mut TxContext) {
18    let poll = Poll {
19      id: object::new(CTX),
20      question: string::utf8(b"Should we adopt this new policy?"),
21      yes_count: 0,
22      no_count: 0,
23      end_ms: DEADLINE_MS,
24    };
25    transfer::share_object(poll);
26  }
27
28  public fun vote_yes(poll: &mut Poll, clock: &Clock) {
29    let now = clock::timestamp_ms(clock);
30    assert(!now <= poll.end_ms, 1);
31    poll.yes_count = poll.yes_count + 1;
32  }
33 }
```

CodeStatistics

SourceBytecode

```
1 module @x6fee75f286c448ec4485ac3c1ff61708d5ce753ae300f128ebca7e06cd:
2   struct Poll has key {
3     id: @x2::object::UID,
4     question: @x1::string::String,
5     yes_count: u64,
6     no_count: u64,
7     end_ms: u64,
8   }
9
10  public fun new_until_2025_12_31_jst(arg0: &mut @x2::tx_context::TxContext) {
11    let v0 = Poll {
12      id: @x2::object::new(arg0),
13      question: @x1::string::utf8(b"Should we adopt this new policy?"),
14      yes_count: 0,
15      no_count: 0,
16      end_ms: 1767193199999,
17    };
18    @x2::transfer::share_object<Poll>(v0);
19  }
20
21  public fun vote_no(arg0: &mut Poll, arg1: &@x2::clock::Clock) {
22    assert!(@x2::clock::timestamp_ms(arg1) <= arg0.end_ms, 1);
23    arg0.no_count = arg0.no_count + 1;
24  }
25
26  public fun vote_yes(arg0: &mut Poll, arg1: &@x2::clock::Clock) {
27    assert!(@x2::clock::timestamp_ms(arg1) <= arg0.end_ms, 1);
28    arg0.yes_count = arg0.yes_count + 1;
29  }
```

Execute

new_until_2025_12_31_jst

vote_no

vote_yes

Arg0

Ox9db091364f7fbbeda18ecc484fd8e54cd226431

Arg1

Ox0

clockオブジェクトは0x0固定です。

Execute