forgeでのEthereumコントラクトデプロイ

Ethereumのローカル開発環境#3669d0

他にもやり方はあると思うが、ここでは `forge script` コマンドでデプロイする。

forgeプロジェクトのトップに `deployments.toml` があり、 `script/Deploy.s.sol` にデプロイ用のSolidityコードがある。実際は `script/Deploy.s.sol` の中で `src/` にあるSolidityコードを `import` しているそのあたりはforgeのサイトを参考にすると良い。

`deployments.toml` は `script/Deploy.s.sol` の中で `_loadConfig()` が読み込んでいる。TOMLファイルはネットワークごとにセクションを用意する。ChainIDの数値かエイリアス名を書き、 `forge script --chain <ID>` で使用するのが安全。 `--chain` がない場合には `--rpc` の方でマッチングするらしいが名前を指定したほうが無難だと思う。
`script/Deploy.s.sol` で `_loadConfig()` したデータはパラメータ名を添字にしてアクセスする。例えばここでは `[anvil.uint]` に `initial_supply` があり、それを `config.get("initial_supply").toUint256()` などとして扱っている。
チェーンごとなどの設定は `deployments.toml` をスクリプトで作成するなりシンボリックリンクで差し替えするなりがよいと思う。

`deployments.toml` の `endpoint_url` は必要なようだし、 `forge` コマンドでの `--rpc-url` も必要だった。

```
forge
    -f, --fork-url <URL>
            Fetch state over a remote endpoint instead of starting from an empty
  state.

            If you want to fetch state from a specific block number, see --fork-
  block-number.

            [aliases: --rpc-url]
```

anvil

```
deployments.toml
  [anvil]
```

```
endpoint_url = "http://127.0.0.1:8545"

[anvil.address]
signers = [
    "0x43500d39a111E5Ccf83C7C1A421AE6bB31464CEE",
    ...略...
]

[anvil.uint]
initial_supply = 1000000000
pegin_fee = 1000
...略...
```

script/Deploy.s.sol

```
// SPDX-License-Identifier: UNLICENSED
pragma solidity ^0.8.30;

import {Script} from "forge-std/Script.sol";
import {Config} from "forge-std/Config.sol";

contract MyScript is Script, Config {
    function run() public {
        _loadConfig("./deployments.toml", true);
        uint256 initialSupply = config.get("initial_supply").toUint256();
        ...略...
```

forge

```
CHAIN_ID="anvil"
RPC="http://127.0.0.1:8545"
PRIVKEY="0x......."
OPTION="--skip-simulation"

forge build
forge script script/Deploy.s.sol --chain "$CHAIN_ID" --rpc-url "$RPC" --broadcast
--private-key "$PRIVKEY" $OPTION
```

Sepolia testnet

現役のSephoilaだが2026年末には終わるらしい。次世代はHoodiという名前。

chainIDは `11155111` 。
アカウントなしで使用できるRPCがいくつかある。今回は以下を使った。
https://ethereum-sepolia-rpc.publicnode.com
Explorerサイトはここでよいだろう。
https://sepolia.etherscan.io/

トランザクションの展開にこのくらいのガス代が必要だった。タイミングによるのかもしれないが目安として。
0.000122ether

bridgeの実装

anvilを使っていた場合は毎回デプロイからし直していたので問題にならなかったが、testnetを使う場合は既にデプロイは終わっているのでノードのウォレットもある程度済んでいるはずだ。
ノード個別の情報には以下があるはずである。

- Bitcoinウォレットのxpub(external, internal)
  - Bitcoinマルチシグウォレット
    `btc-xpub-***.txt` の1行目を読んで末尾の `/0/*` を削除した文字列を使い、externalは `/0/*` を、internalは `/1/*` を付けて使う。
    記号類が入っているのでそのままコマンドライン引数には使いづらい。
- Ethereumアカウント
  - ERC4337マルチシグ
- P2Pノード
  - スロットの順番

これまで追加したことがあるノードの情報はEthereum側に保存しておくべきかもしれない。そしてどれか1つでも当てはまる場合には追加できないようにすべきだろう。