Smart contract development with TokenSPICE & Brownie

PyCon Iran, Feb 16, 2022

Trent McConaghy
@trentmc0 @oceanprotocol

http://trent.st/content/pycon.pdf

ocean

Overview

Who this talk is for:

- Py / ML developers
- Who are blockchain-curious
- Who could quickly become blockchain ninjas but don’t know it yet!

ocean

Outline

- Ninja strategy #1: a useful cheat

- Smart contracts background
- Ninja strategy #2: for rest of talk

- Py skills →Brownie
- ML algs → blockchain algs
- Py + ML skills →TokenSPICE

ocean

Ninja Strategy #1:
A useful cheat

ocean

Py + ML without Solidity dev, for ML use cases

Tokenize data & algorithms, share it, sell it (Ocean)

github.com/oceanprotocol/ocean.py

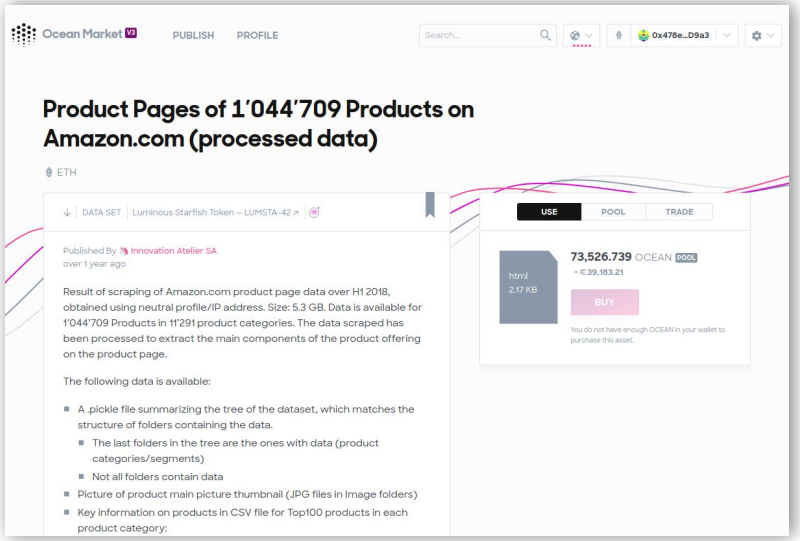
Publish datatokens

In the Python console:

```python
import os
from ocean_lib.example_config import ExampleConfig
from ocean_lib.ocean.ocean import Ocean
from ocean_lib.web3_internal.wallet import Wallet

private_key = os.getenv('TEST_PRIVATE_KEY1')
config = ExampleConfig.get_config()
ocean = Ocean(config)

print("create wallet: begin")
wallet = Wallet(ocean.web3, private_key, config.block_confirmations, config.transaction_timeout)
print(f"create wallet: done. Its address is {wallet.address}")

print("create datatoken: begin.")
datatoken = ocean.create_data_token("Dataset name", "dtsymbol", from_wallet=wallet)
print(f"created datatoken: done. Its address is {datatoken.address}")
```

Congrats, you've created your first Ocean datatoken! 🐳

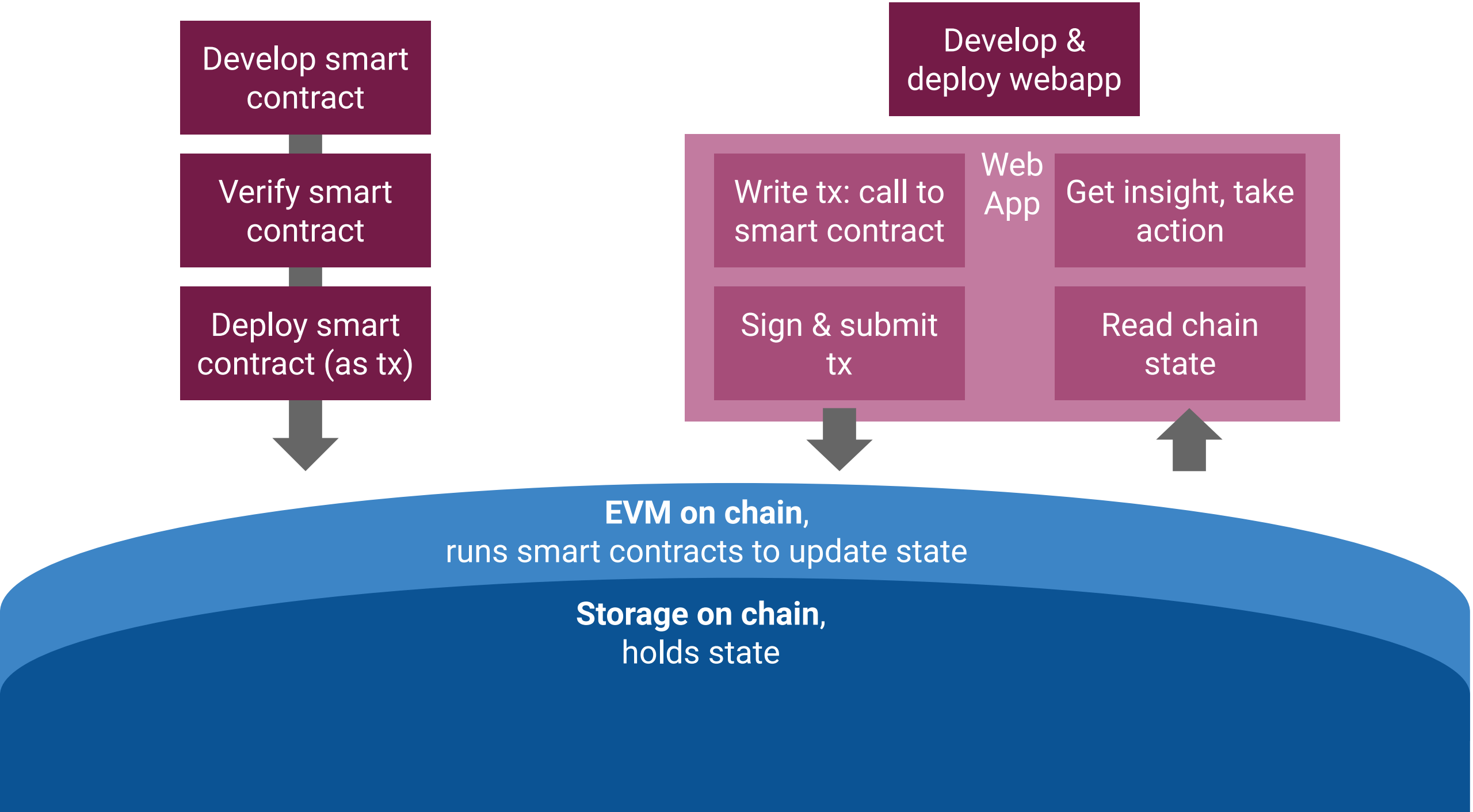
[market.oceanprotocol.com](https.://market.oceanprotocol.com)



ocean

Background on Smart Contracts

ocean


Develop smart contract
Verify smart contract
Deploy smart contract (as tx)
Develop & deploy webapp
Web App
Write tx: call to smart contract
Get insight, take action
Sign & submit tx
Read chain state
EVM on chain,
runs smart contracts to update state
Storage on chain,
holds state

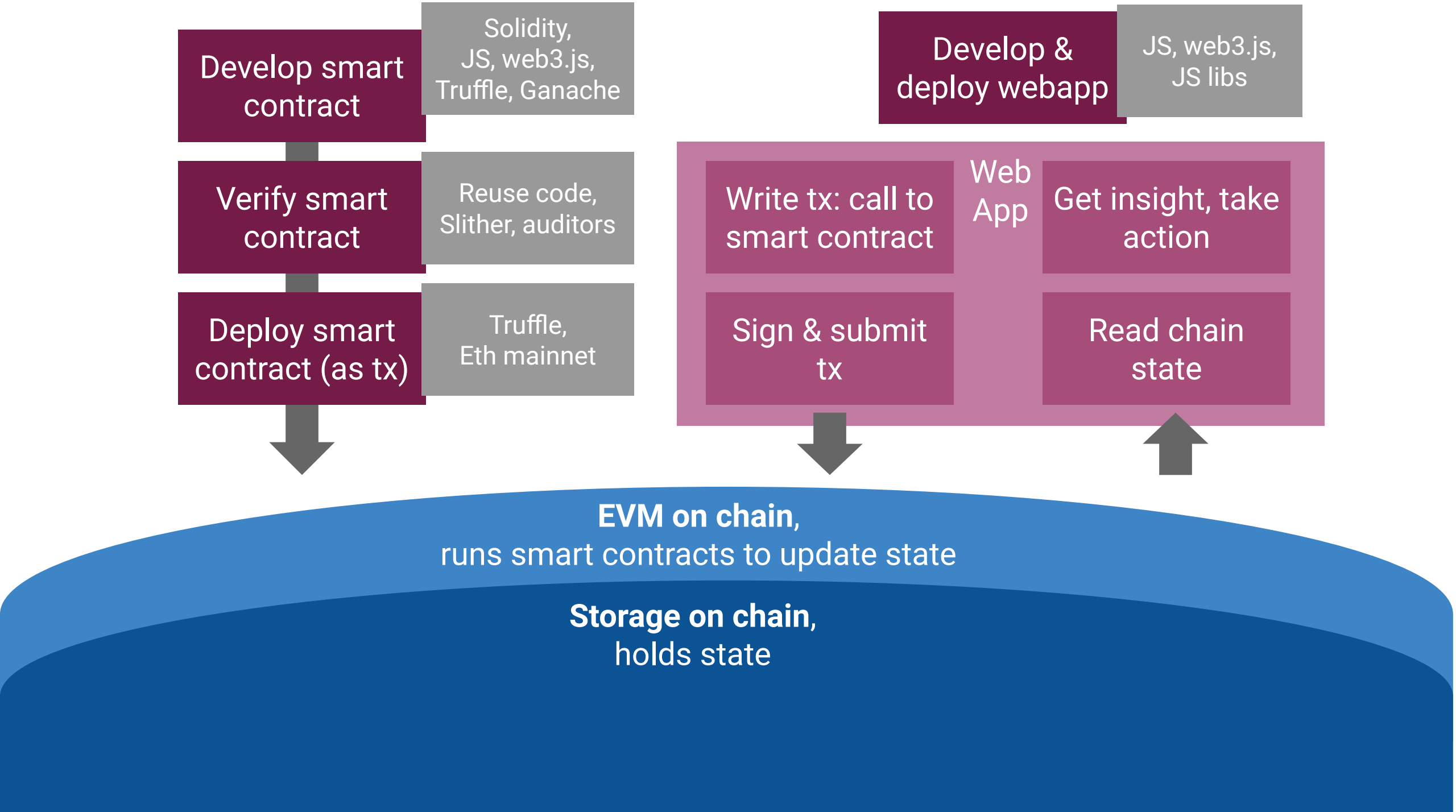
What a smart contract looks like (Solidity code)

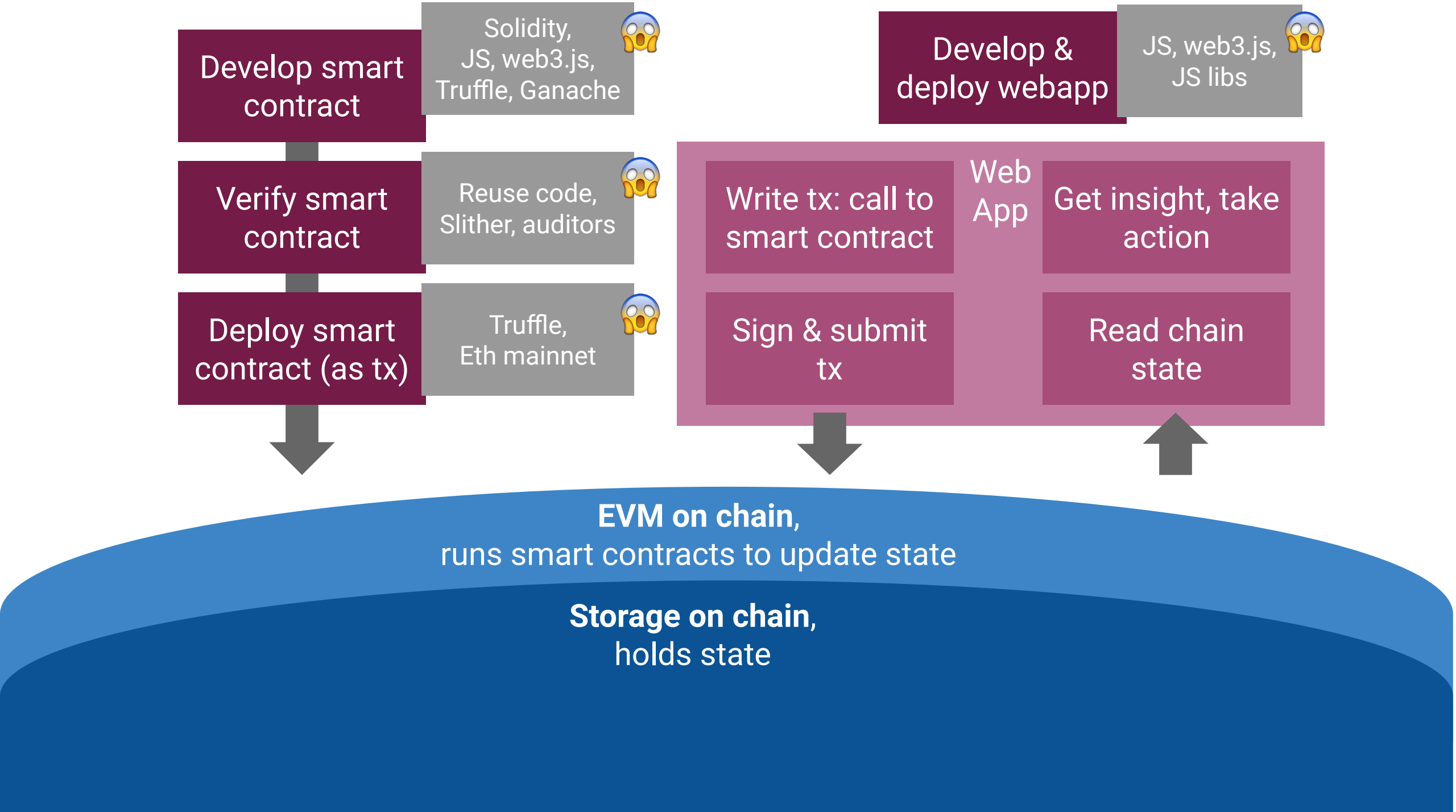
```
function createToken(
    string memory blob,
    string memory name,
    string memory symbol,
    uint256 cap
)
    public
    returns (address token)
{
    require(
        cap != 0,
        'DTFactory: zero cap is not allowed'
    );

    token = deploy(tokenTemplate);

    require(
        token != address(0),
        'DTFactory: Failed to perform minimal deploy of a new token'
    );
    IERC20Template tokenInstance = IERC20Template(token);
    require(
        tokenInstance.initialize(
            name,
            symbol,
            msg.sender,
            cap,
            blob,
            communityFeeCollector
        ),
        'DTFactory: Unable to initialize token instance'
    );
    emit TokenCreated(token, tokenTemplate, name);
```

```
/**
 * @dev mint
 *      Only the minter address can call it.
 *      msg.value should be higher than zero and gt or eq minting fee
 * @param account refers to an address that token is going to be minted to.
 * @param value refers to amount of tokens that is going to be minted.
 */
function mint(
    address account,
    uint256 value
)
    external
    onlyMinter
{
    require(
        totalSupply().add(value) <= _cap,
        'DataTokenTemplate: cap exceeded'
    );
    _mint(account, value);
}
```

https://github.com/oceanprotocol/contracts

ocean


Develop smart contract
Solidity, JS, web3.js, Truffle, Ganache
Verify smart contract
Reuse code, Slither, auditors
Deploy smart contract (as tx)
Truffle, Eth mainnet
Develop & deploy webapp
JS, web3.js, JS libs
Web App
Write tx: call to smart contract
Get insight, take action
Sign & submit tx
Read chain state
EVM on chain, runs smart contracts to update state
Storage on chain, holds state


Develop smart contract
Solidity, JS, web3.js, Truffle, Ganache
Verify smart contract
Reuse code, Slither, auditors
Deploy smart contract (as tx)
Truffle, Eth mainnet
Develop & deploy webapp
JS, web3.js, JS libs
Web App
Write tx: call to smart contract
Get insight, take action
Sign & submit tx
Read chain state
EVM on chain,
runs smart contracts to update state
Storage on chain,
holds state

Motivation

Why blockchain could look daunting to MLers:

- Different languages: Solidity, JS
- Different tools: Web3.js, Truffle, Ganache, ..
- Different building blocks: ERC20, ERC721, AMMs, multisig, DAOs, ..

Extra Worries:

- Is it start from zero? It looks like a long / steep learning curve
- Is it worth it? It looks like building webapps, not ML algorithms. Different style.

ocean

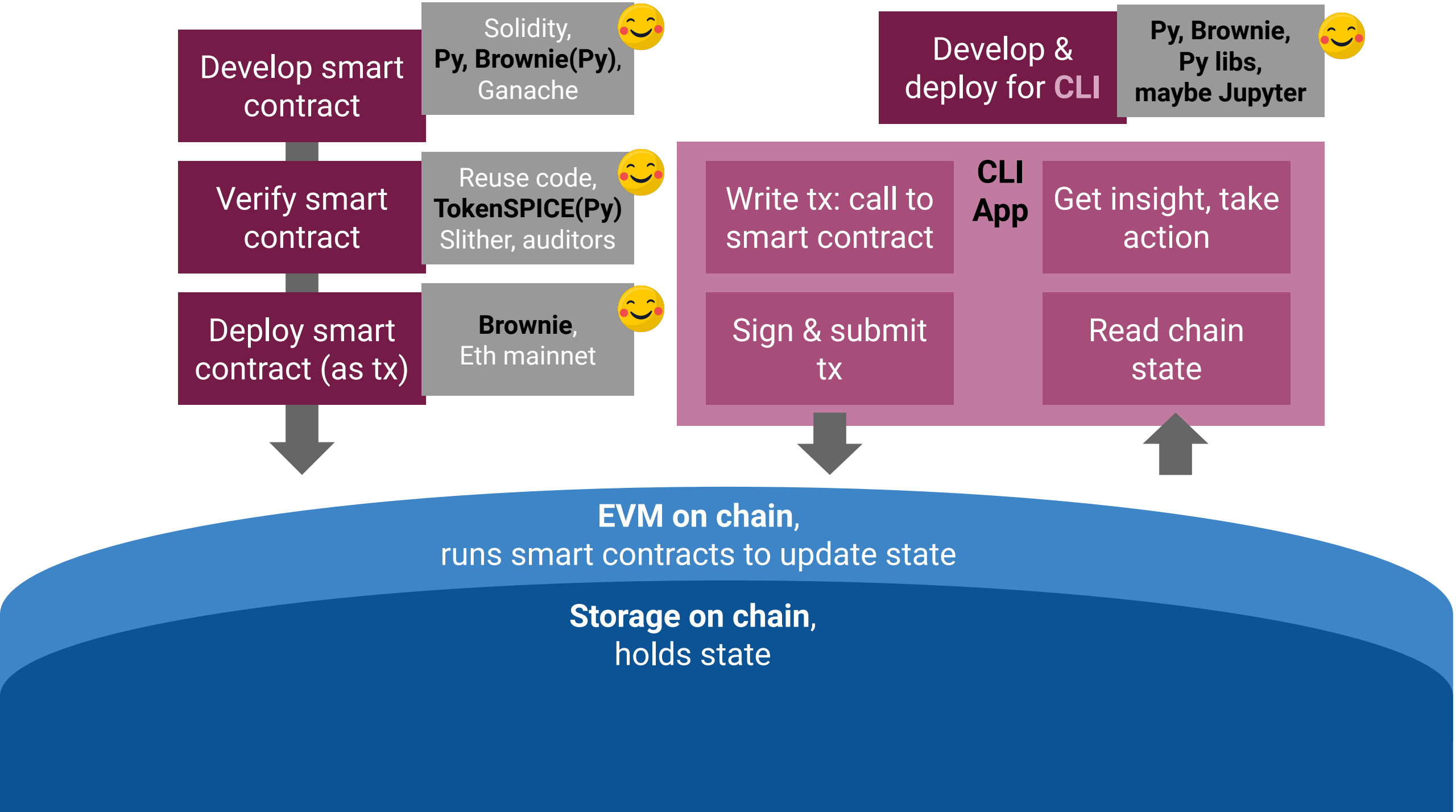
Ninja Strategy #2:
Strategy for rest of talk

What if…

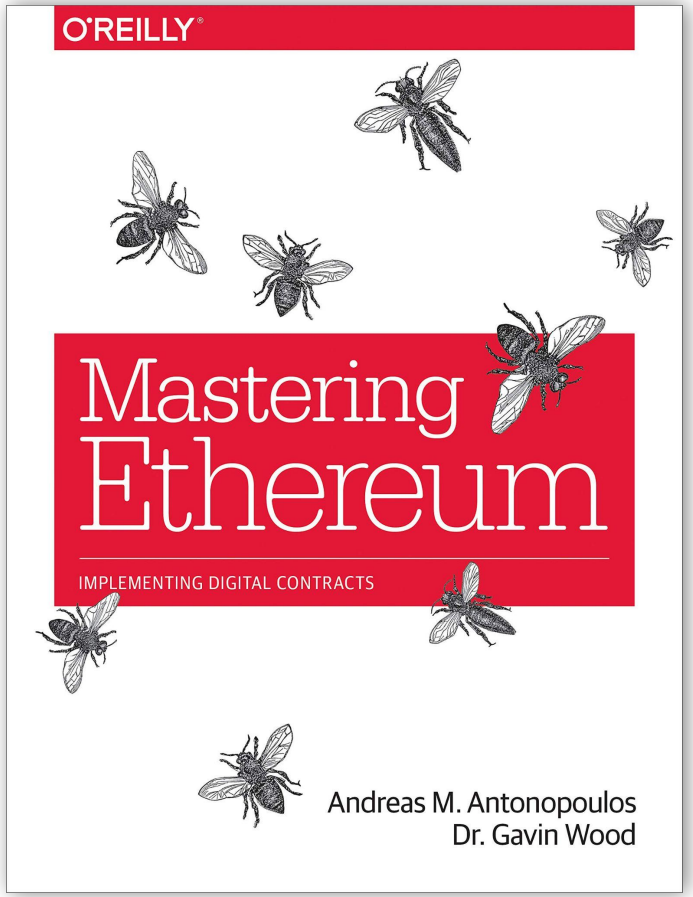
- Solidity, JS → **Mostly Py**, some Solidity
- Web3.js, Truffle, Ganache → **Brownie (Py)**, Ganache (but hidden)
- ERC20, ERC721, AMMs, multisig, DAOs → **treat as Py classes/objects: Brownie**

- Start from zero → **Py & ML ninja skills are your big lever**
- Webapps, not ML-like algs → **ML-like algs via TokenSPICE (Py)**

ocean


Develop smart contract
Solidity, Py, Brownie(Py), Ganache
Verify smart contract
Reuse code, TokenSPICE(Py) Slither, auditors
Deploy smart contract (as tx)
Brownie, Eth mainnet
Develop & deploy for CLI
Py, Brownie, Py libs, maybe Jupyter
CLI App
Write tx: call to smart contract
Get insight, take action
Sign & submit tx
Read chain state
EVM on chain, runs smart contracts to update state
Storage on chain, holds state

Learning Solidity



You still need Eth & Solidity basics.

This is the best path to a solid foundation.

amazon.com/Mastering-Ethereum-Building-Smart-Contracts/dp/1491971940/

And, JS not needed 😊

ocean

Py skills →Brownie

Recall: what if…

- Solidity, JS → **Mostly Py**, some Solidity
- Web3.js, Truffle, Ganache → **Brownie (Py)**, Ganache (but hidden)
- ERC20, ERC721, AMMs, multisig, DAOs → **treat as Py classes/objects: Brownie**

ocean

Brownie Quickstart

Let’s walk through **“Getting Started With Brownie”**

Part 1 - Install

Part 2 - Brownie projects

Part 3 - Basic functionality

1: https://iamdefinitelyahuman.medium.com/getting-started-with-brownie-part-1-9b2181f4cb99

2: https://betterprogramming.pub/getting-started-with-brownie-part-2-615a1eec167f

3: https://betterprogramming.pub/getting-started-with-brownie-part-2-615a1eec167f

ocean

ML Algs → Blockchain Algs

Recall: what if…

- Webapps, not ML-like algs → **ML-like algs**

ocean

Incentives & Blockchains

“Show me the incentive, and I will show you the outcome”
-- Charlie Munger

Incentives are conceptually easy in blockchain:
Get people to do stuff, by paying them in tokens.

How to implement incentives in blockchains:
Develop, verify, and deploy Solidity code

ocean

From ML Algorithm Design To Incentive Design

How do we *design* the incentives?

This problem is a *lot* like ML algorithm design:

It's an optimization problem formulation!

Minimize fi(x)

S.t. gj(x) <= 0

And hk(x) = 0

This is design of analog / continuous-valued systems, vs digital / discrete.

In blockchain land, incentive design = **Token Engineering**.

ocean

Verification

How do we *verify* the incentives?

Three ways:

1. Manual → human feedback
2. Economic → deploy live, ratchet up risk
3. Software-based → need appropriate SW

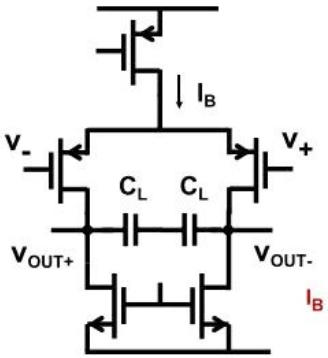
ocean

Verifying Continuous-Valued Systems: Analog Circuits

Design analog circuit

Verify with SPICE simulator

(Manufacture circuit)



$$GBW = \frac{g_{m1}}{2\pi C_L} \qquad g_{m1} = \frac{I_B}{V_{GS1}-V_T}$$

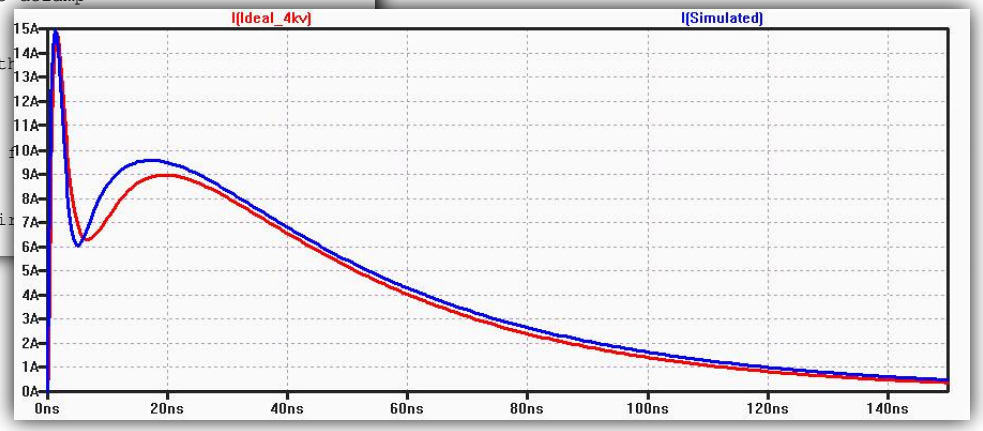
$$GBW_{max} = \frac{I_B}{V_{GS1}-V_T}\,\frac{1}{2\pi C_L}$$

0.2 V

$I_B = 10\ \mu A$ $C_L = 1$ pF $GBW_{max} \approx 10$ MHz [8]

$$FOM = \frac{GBW.C_L}{I_B} = 1000\ \text{MHzpF/mA} \quad [800]$$

```
SPICE sample circuit - diode clamp

*independent voltage source with DC value, AC value, and
*transient square wave. -10V to +20V extent, with 2ms period
V1 1 0 1 AC 1 pulse -10 20 0 1.e-8 1.e-8 1e-3 2e-3

*capacitor for clamping
C1 1 2 1e-6

*diode for clamp - model name is dclamp
D1 2 0 dclamp

*load resistor - large enought t
*model for diode
.model dclamp D(IS=1e-14)

*DC transfer function generated
.DC V1 -20 20 .1

*AC frequency sweep - assumes ci
```



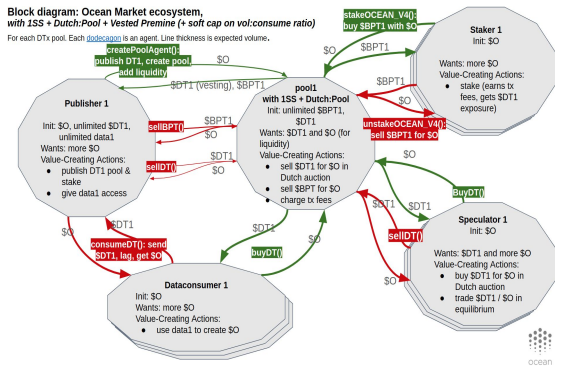
ocean

Verifying Continuous-Valued Systems: Incentives

Develop smart contract

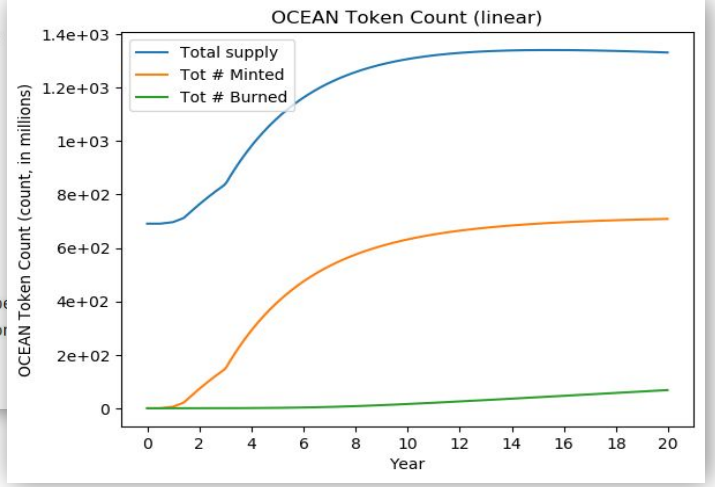
Verify with TokenSPICE simulator

Deploy smart contract (as tx)




```
uint spotPriceBefore = calcSpotPrice(
    inRecord.balance,
    inRecord.denorm,
    outRecord.balance,
    outRecord.denorm,
    _swapFee
);
require(spotPriceBefore <= maxPrice, 'ERR_BAD_LIMIT_PRICE');

tokenAmountOut = calcOutGivenIn(
    inRecord.balance,
    inRecord.denorm,
    outRecord.balance,
    outRecord.denorm,
    tokenAmountIn
```

```
new_agents.add(
    PublisherAgent(
        name="publisher",
        USD=0.0,
        OCEAN=self.ss.publisher_init_OCEAN,
        pub_ss=pub_ss,
    )
)

new_agents.add(
    DataconsumerAgent(
        name="consumer",
        USD=0.0,
        OCEAN=self.ss.consumer_init_OCEAN,
        s_between_buys=self.ss.consumer_s_be
        profit_margin_on_consume=self.ss.con
    )
)
```



ocean

Py + ML skills → TokenSPICE

ocean

Recall: what if…

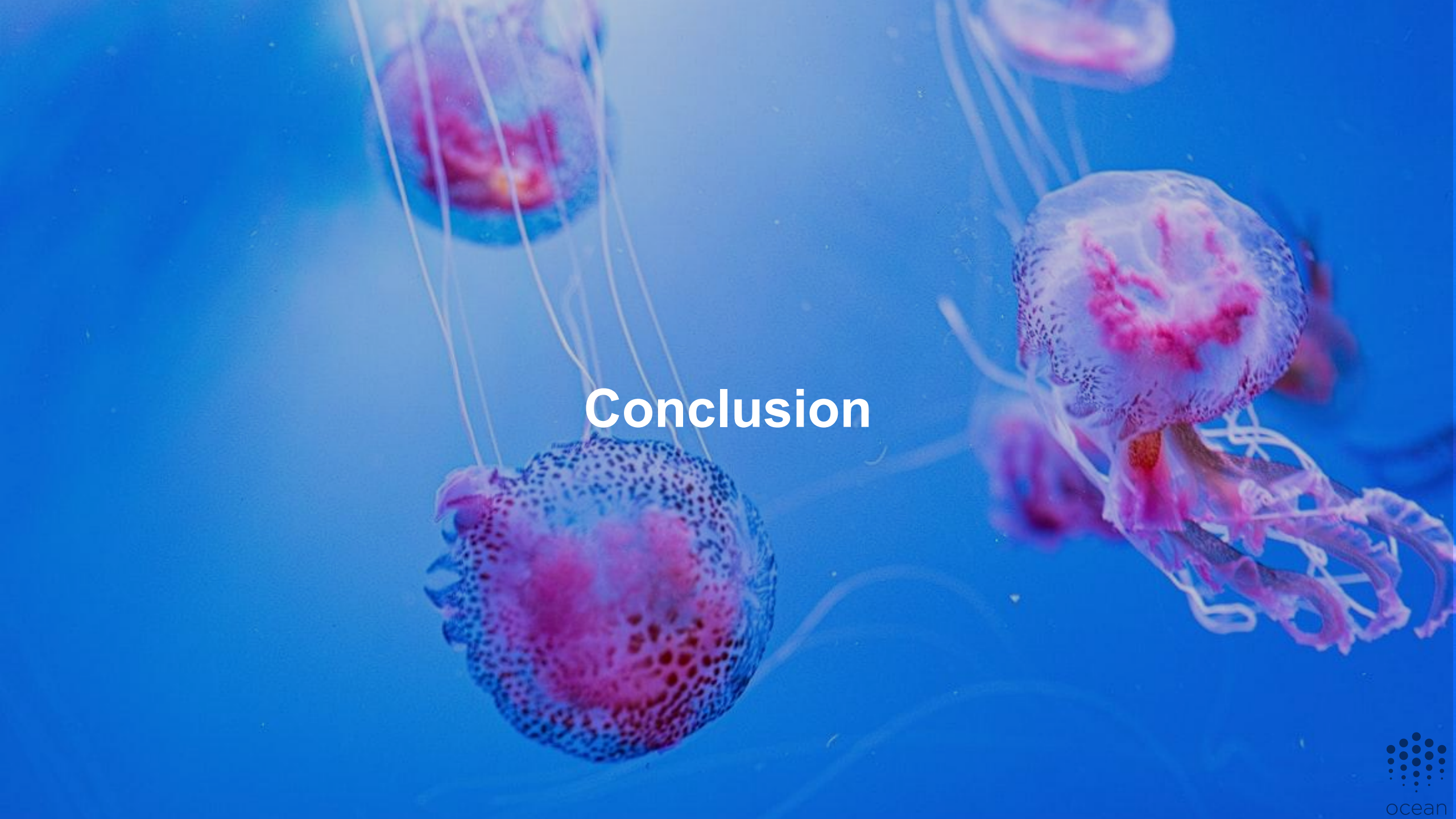
- Start from zero → **Py & ML ninja skills are your big lever**
- Webapps, not ML-like algs → **ML-like algs via TokenSPICE (Py)**

ocean

TokenSPICE Quickstart

Let’s walk through **TokenSPICE’s README**

github.com/tokenspice/tokenspice

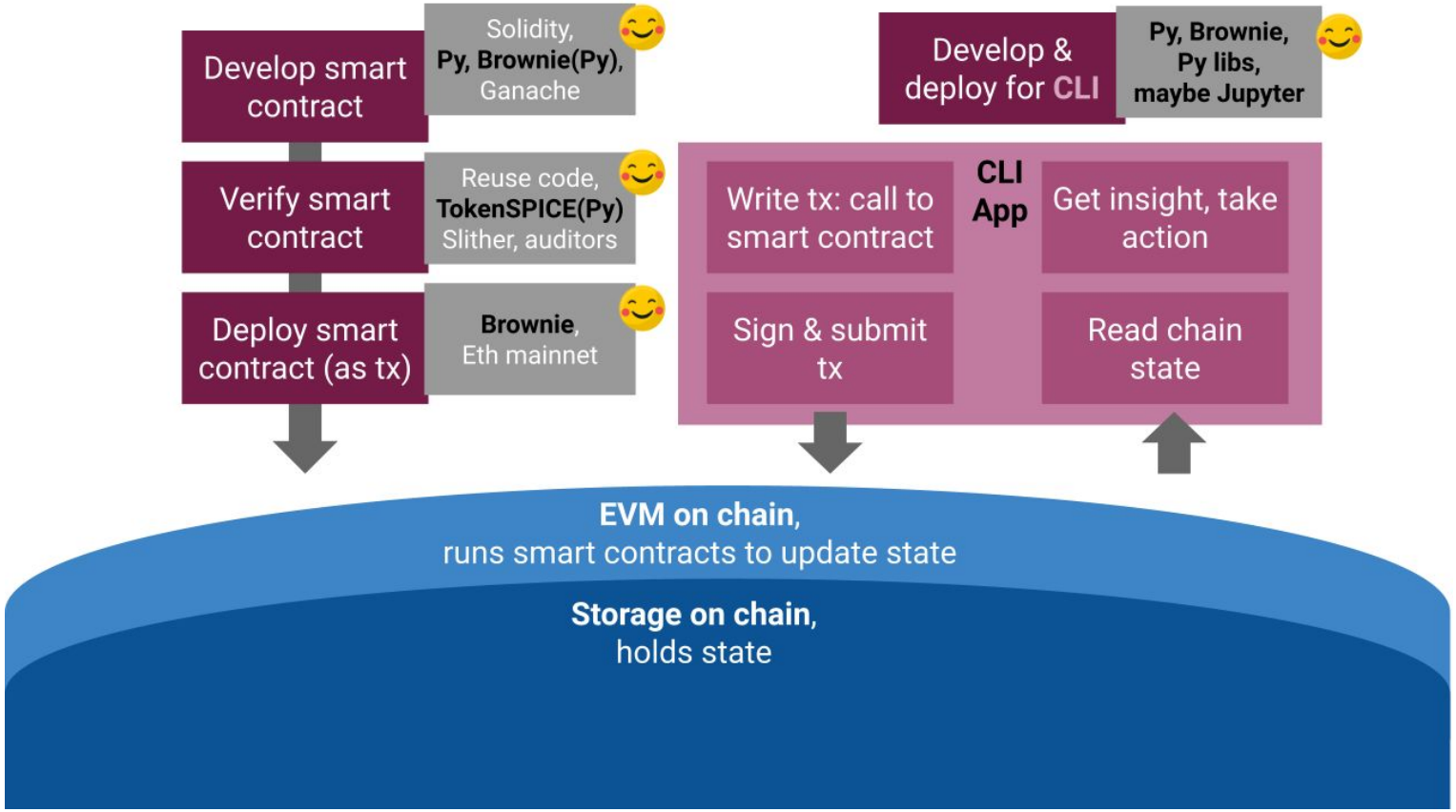


Conclusion

ocean

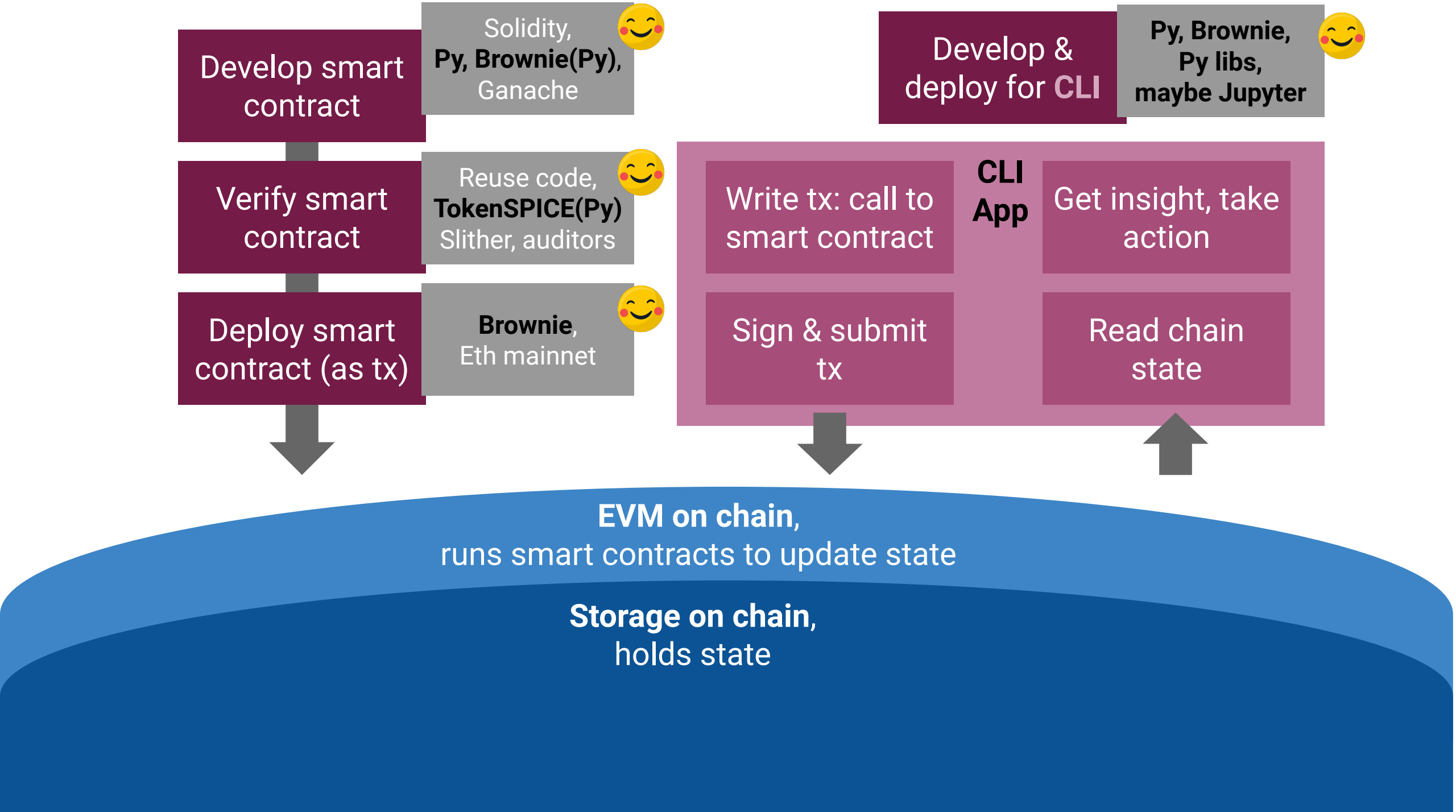
Conclusion

- You know Py + ML, and you want to do cool stuff in blockchain
- Ninja strategy #1: skip Solidity, use ocean.py to tokenize data & algs
- **Ninja strategy #2: dev on Solidity, use Brownie & TokenSPICE**



@trentmc0
@oceanprotocol

ocean


Develop smart contract
Solidity, Py, Brownie(Py), Ganache
Verify smart contract
Reuse code, TokenSPICE(Py) Slither, auditors
Deploy smart contract (as tx)
Brownie, Eth mainnet
Develop & deploy for CLI
Py, Brownie, Py libs, maybe Jupyter
CLI App
Write tx: call to smart contract
Get insight, take action
Sign & submit tx
Read chain state
EVM on chain, runs smart contracts to update state
Storage on chain, holds state