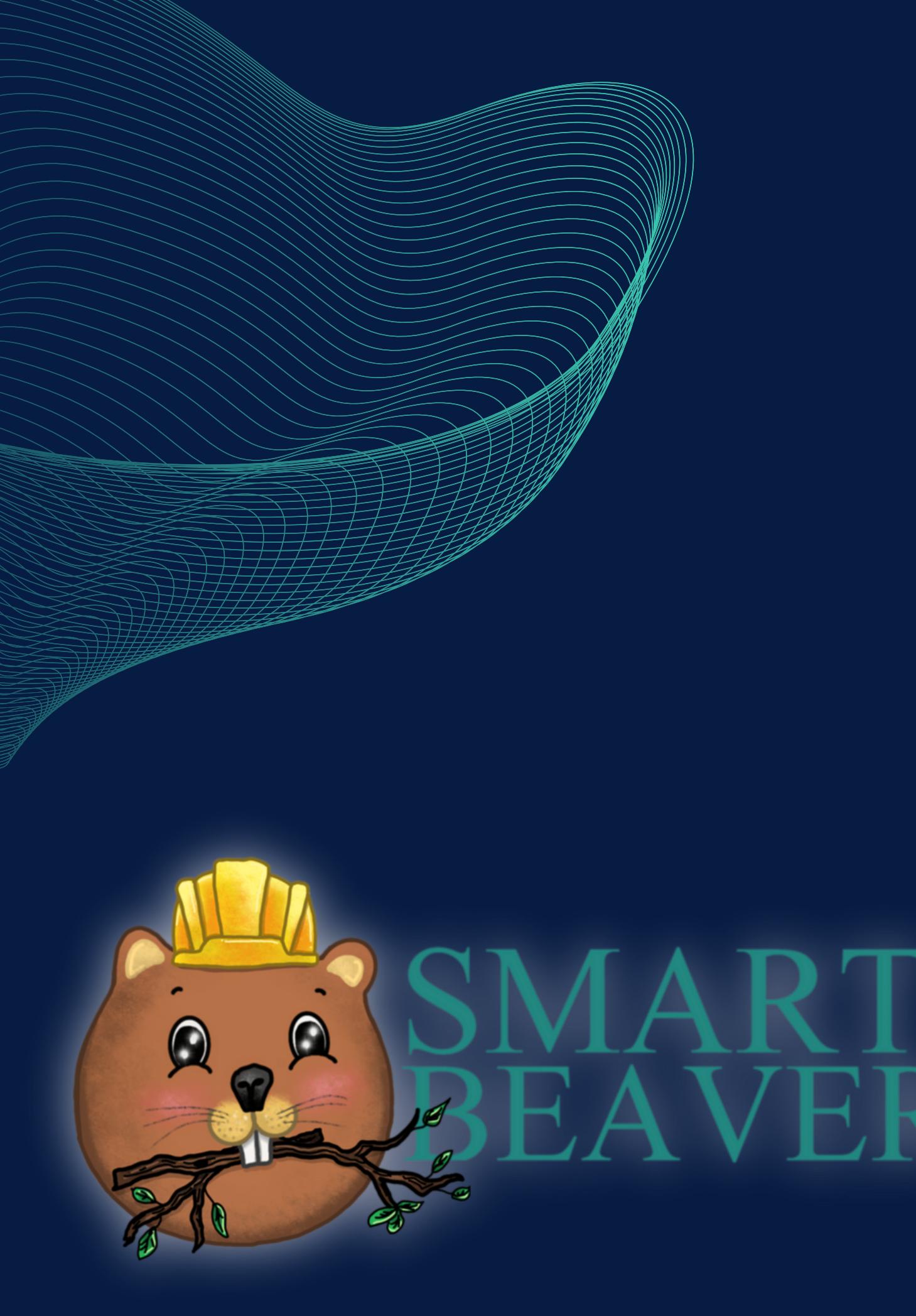


SMART BEAVER

ETHWarsaw23





Are you tired of using OpenBrush? Are you frustrated with their custom library, lack of updates, and limited features?

Do you dream of having a wizard that can easily configure and deploy contracts written in **clean ink!**?

Welcome to Smart Beaver! We specialize in building contracts as **sturdy as dams**.

3 Simple steps



SMART
BEAVER

Configure

Choose the smart contract standard you want to deploy, and then configure its properties

Inspect

Inspect the generated source code; you can choose to deploy it or use it as a template.

Deploy

Connect your wallet and either deploy the contract or download the compiled WASM.

Configure



SMART BEAVER

Log in

PSP22 PSP37 PSP34

Compile code Connect wallet

Settings

Features

- Metadata
- Mintable
- Burnable
- Pausable
- Capped

Access Control

Upgradeability

```
#![cfg_attr(not(feature = "std"), no_std, no_main)]
#[ink::contract]
mod psp22 {
    use ink::storage::Mapping;
    use ink::prelude::{
        vec,
        vec::Vec,
        string::String,
    };

    /// Defines the storage of your contract.
    /// Add new fields to the below struct in order
    /// to add new static storage fields to your contract.
    #[ink(storage)]
    pub struct Psp22 {
        /// Total token supply.
        total_supply: Balance,
        /// Mapping from owner to number of owned token.
        balances: Mapping<AccountId, Balance>,
        /// Mapping of the token amount which an account is allowed to withdraw
        /// from another account.
        allowances: Mapping<(AccountId, AccountId), Balance>,
    }
}
```

Copy

Inspect

The screenshot shows the 'Inspect' interface with a dark theme. At the top, there are three tabs: 'PSP22' (highlighted in green), 'PSP37', and 'PSP34'. To the right are two buttons: 'Compile code' and 'Connect wallet'. On the left, there is a sidebar with the following sections and their current status:

- Settings**: A dropdown menu.
- Features**: An expanded section containing:
 - Metadata
 - Mintable
 - Burnable
 - Pausable
 - Capped
- Access Control**: A dropdown menu.
- Upgradeability**: A dropdown menu.
- Info**: A dropdown menu.

The main area displays the following Ink code:

```
/// Defines the storage of your contract.
/// Add new fields to the below struct in order
/// to add new static storage fields to your contract.
#[ink(storage)]
pub struct Psp22 {
    /// Total token supply.
    total_supply: Balance,
    /// Mapping from owner to number of owned token.
    balances: Mapping<AccountId, Balance>,
    /// Mapping of the token amount which an account is allowed to withdraw
    /// from another account.
    allowances: Mapping<(AccountId, AccountId), Balance>,
    owner: AccountId,
    is_paused: bool,
    cap: u128,
    pub name: Option<String>,
    pub symbol: Option<String>,
    pub decimals: u8,
}

#[ink::trait_definition]
pub trait Psp22Burnable {
    // Burns `amount` of tokens from `account`
    #[ink(message)]
    fn burn(&mut self, account: AccountId, amount: Balance) -> PSP22Result<()>;
}
```

A 'Copy' button is located in the top right corner of the code editor area.

Contracts in clean ink!

Compile

TAVER

PSP22 PSP37 PSP34

Compile code Download compiled contract bundle Deploy contract ⚡ Disconnect

Settings

Features

Access Control

Upgradeability

Info

Submit

Contract building in progress. Please wait

Copy

```
/// Defines the storage of your contract.  
/// Add new fields to the below struct in order  
/// to add new static storage fields to your contract.  
#[ink(storage)]  
pub struct Psp22 {  
    /// Total token supply.  
    total_supply: Balance,  
    /// Mapping from owner to number of owned token.  
    balances: Mapping<AccountId, Balance>,  
    /// Mapping of the token amount which an account is allowed to withdraw  
    /// from another account.
```

Deploy

4

Compile code Download compiled contract bundle Deploy contract

```
#![cfg_attr(not(feature = "std"), no_std, no_main)]
#[ink::contract]
mod psp22 {
    use ink::storage::Mapping;
    use ink::prelude::{
        vec,
        vec::Vec,
        string::String,
    };

    /// Defines the storage of your contract.
    /// Add new fields to the below struct in order
    /// to add new static storage fields to your contract.
    #[ink(storage)]
    pub struct Psp22 {
        /// Total token supply.
        total_supply: Balance,
        /// Mapping from owner to number of owned token.
        balances: Mapping<AccountId, Balance>,
        /// Mapping of the token amount which an account is allowed to withdraw.
    }
}
```

Details

From AlephZero

Network Aleph Zero Testnet

Fees 0.004682253728 TZERO

Tip 0 TZERO

Method contracts : instantiateWithCode

Description Instantiates a new contract from the supplied `code` optionally transferring

Arguments as : YAML / JSON

```
value: '0'
gas_limit:
refTime: 10,000,000
proofSize: '0'
storage_deposit_limit: null
```

Close



Currently supported

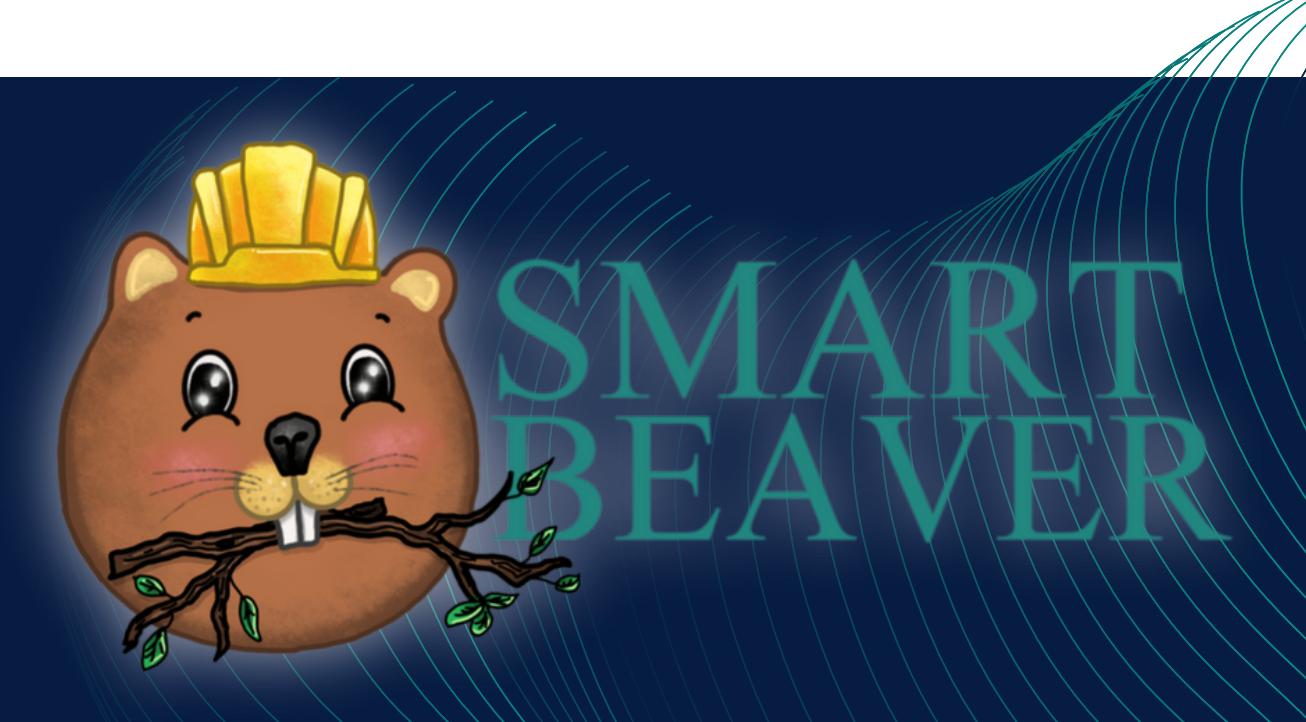
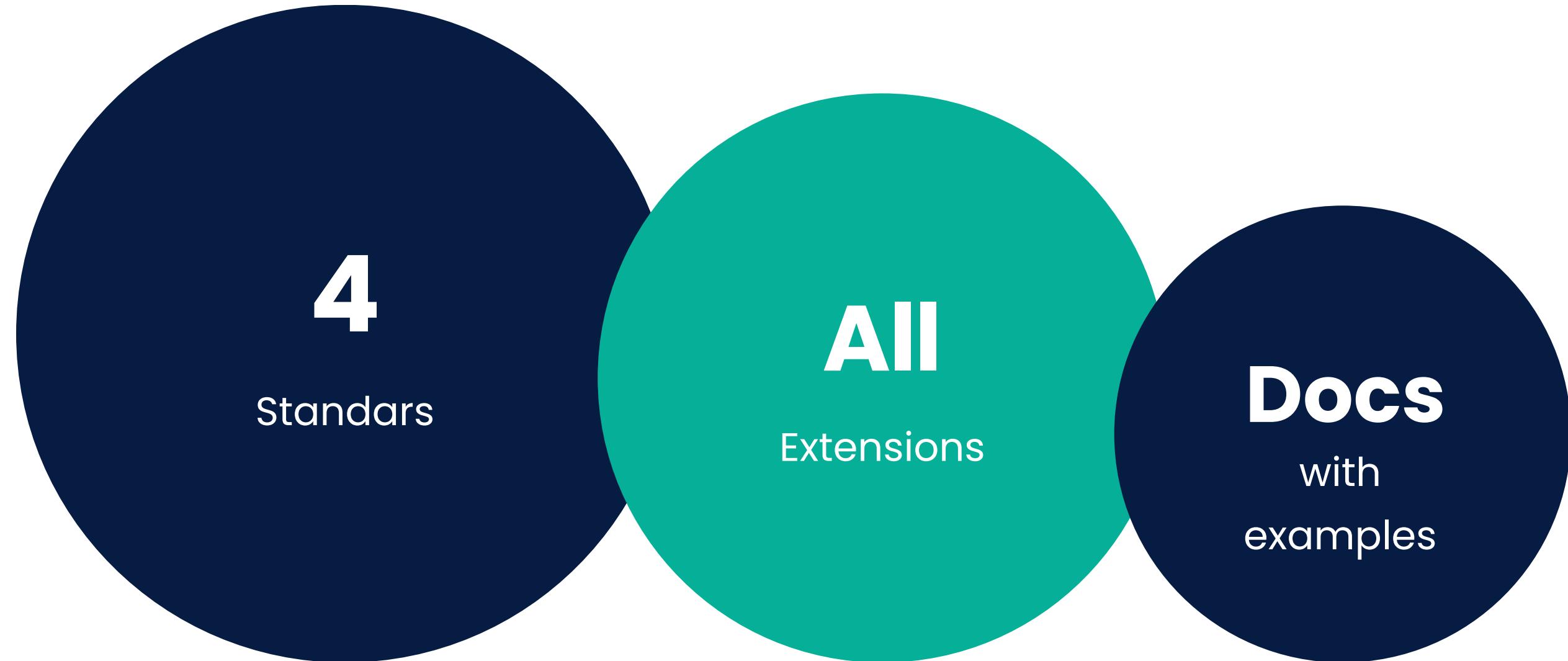
1 STANDARD

5 EXTENSIONS

3 SETTINGS



Our future



Smart Beaver



Izabela Łaszczuk

Maciej Malik

Piotr Swierzy