How to add contract templates

Introduction

Once you have successfullycreated your project ↗ , you can addcontract templates .

Acontract template is a pre-defined, reusable blueprint or pattern for creating a smart contract in a blockchain environment. It provides a structured starting point with predefined functionalities, variables, and logic that can be customized to suit specific use cases.

i A contract template is a specific implementation that can be added to a project. These templates serve as a foundation for creating contracts by providing pre-configured settings, initial code, and common functionalities. They streamline the process of developing contracts by offering a structured starting point that developers can build upon. By using contract templates, developers can save time and effort in setting up the basic structure and functionalities of their contracts, allowing them to focus on customizing and implementing the specific features required for their use case.

Overview

If you want to add a template to a contract, you first need to navigate into your project's directory and run the following command:

jenesis add contract You can find all the contract templates available inJenesis Templates ↗(opens in a new tab) GitHub repository. As for an overview, the available templates are the following:

- starter
- : a bare-bones contract that sets and increments a counterhere ↗(opens in a new tab)
- .
- token
- : a fungible token contracthere ↗(opens in a new tab)
- .
- proxy
- : a basic proxy contracthere ↗(opens in a new tab)
- multisig
- : a fixed multisig contracthere ↗(opens in a new tab)
- .
- nft
- : a general token contract supporting nftshere ↗(opens in a new tab)
- .
- oracle
- : a contract storing a value to be requested for a feehere ↗(opens in a new tab)
- .
- oracle-client
- : a contract for requesting a value for a fee from a specified oracle contracthere ↗(opens in a new tab)
- .

Example: CW-20 basic template for fungible token contract

Thecw20-base template is designed for creating CW-20 tokens on the Fetch.ai blockchain. CW-20 tokens are a standard interface for fungible tokens. This template provides a solid starting point for developers looking to create their own custom CW-20 tokens, complete with essential functionalities like transfers, approvals, and querying balances.

The steps needed to add thecw20-base template with the namemy_token are provided below:

jenesis add contract cw20-base my_token If you need multiple deployments of the same contract, you can use the--deployments or-d flag to specify multiple deployments and name them:

jenesis add contract [--deployments] Jenesis will add the deployments to all profiles for the specified contract.

In the example below,token_1 andtoken_2 deployments have been added. This will allow you to deploymy_token contract with two different configurations. You can add as many deployments as we wish:

jenesis add contract cw20-base my_token -d token_1 token_2 If no deployments are selected when adding a contract, the default deployment name will be equal to the contract name.

Theadd contract command will add a contract template to our jenesis project insidecontracts/my_token/ folder. It will also update thejenesis.toml configuration file with the contract information:

[profile.testing.contracts.token_1] name =

"token_1" contract =

"my_token" network =

"fetchai-testnet" deployer_key =

"" init_funds =

""

[profile.testing.contracts.token_2] name =

"token_2" contract =

"my_token" network =

"fetchai-testnet" deployer_key =

"" init_funds =

""

[profile.testing.contracts.token_1.init]

[profile.testing.contracts.token_2.init] Thedeployer_key field can be manually specified. You can choose any private key locally available to deploy any specific contract. You can also leave this field empty since thedeploy command has an optional argument to deploy all contracts inside a specified profile with the same key, overriding thisdeployer_key argument in thejenesis.toml file. Checkout thedeploy contracts ↗ guide for more information.

Finally, theinit section contains the parameters needed in the instantiation message for this contract to be deployed. The required parameters are taken from the schema file inside thecontracts directory. Since this contract template doesn't include a schema, it will be generated whencompiling ↗ themy_token contract loading the init fields to thejenesis.toml file. You will need to manually add the values for these parameters in their correct variable type, which are listed on the schema file. For thismy_token contract, you need to fill theinit fields for each deployment aftercompiling ↗ .

Below, we can find an example for this:

[profile.testing.contracts.token_1.init] decimals = 6 name = "my_token_name" symbol = "SYMBOL" initial_balances = [{ address = "fetch1d25ap9danl4726uk2nt307y630v87h3h2vq6pl", amount = "5000"}]

[profile.testing.contracts.token_2.init] decimals = 6 name = "my_token_name_2" symbol = "SYMBOL" initial_balances = [{ address = "fetch1d25ap9danl4726uk2nt307y630v87h3h2vq6pl", amount = "2000"}] If your contract requires nested instantiation messages, you may add fields following this structure:

[profile.testing.contracts.example-nested-contract.init] price = {amount = 1000, denom = DLS} info = {performance = {max_speed = 200, unit = kph}, fuel = {consumption = 7, unit = kmpl}} ⚠ Before editing thejenesis.toml configuration file with the desireddeployer_key andinit parameters, make sure to firstcompile ↗ your contract. All configuration parameters will restart every time a contract is compiled if the schema has changed You can also add contracts manually by copying and pasting the contract directory from another project you may have. However, they would need to follow the same directory structure as thestarter ↗(opens in a new tab) template.

When adding a contract manually, you need to update thejenesis.toml file with the contract information by running the following command:

jenesis update Theupdate command will automatically detect which contract is missing in thejenesis.toml configuration file by revising the contracts directory.

Add contract deployments

You can also add further deployments for a given contract by specifying the contract name and the deployment name.

If you want to add a third token calledtoken_3 usingmy_token contract, you can run:

jenesis add deployment my_token token_3 This will automatically create another deployment entry calledtoken_3 .

Attach deployed contracts

If you add a contract into the project's contract folder that has already been deployed in the network, you can attach the deployment to your project for future interaction using theattach command.

You can add a deployment to your project by running the following command:

jenesis add contract cw20-base my_token -d token_1 Then, you would need to compile the contract:

jenesis compile You can also attach the contract. In this case, you will need to specify the deployment's name and address. You can optionally specify the profile where you wish to insert the contract into. If this is not specified, the deployment will be attached to the default profile, which is the first profile created in our project, unless thedefault settings are manually changed:

jenesis attach token_1 fetch18xs97q6h9zgh4sz730a42pp0dqa9sh4eef7eutfkv69q3v2y3x8s72pkua This will add the relevant deployment information into ajenesis.lock file and you will now be able to interact withtoken_1 usingcontract interactions ↗ .

Was this page helpful?

How to add profiles How to compile contracts