Host Env (+ SDK) Proposal

PR: https://github.com/stellar/rs-soroban-env/pull/1355

Allow Extending Instance and Code TTL With Separate Values on the Host Environment For More Cost-Efficient Implementations

tommaso@stellar.org, originated from https://discord.com/channels/897514728459468821/1207091113475706880/1207246458655346718

This proposal aims to introduce a change within the soroban host environment and SDK to enable the guest environment to bump the instance and the contract code with different ttl and threshold values. The goal of this proposal is to enable for better distribution of rent costs among binaries designed to be associated with more than one instance.

Motivation

Currently, when bumping/extending ttl of the instance, the guest environment can call a host function that will bump by the same amount both the contract instance entry and the linked contract code entry. In decentralized contracts, the contract itself bumps its lifetime from within the code with certain thresholds with the idea of costs being distributed among users.

Extending the life for contract code entries is between the most expensive costs of an invocation due to the large size the binary occupies on the ledger, and there are numerous situations where a contract code entry is referenced by more than one instance, thus not allowing to bump separately the contract code and the instance prevents implementing a more efficient lifetime bump logic.

For example, think of a liquidity pool contract, used by hundreds (or thousands) of actively bumped contract instances. The instance of a single pool contract is bumped by all the users of that contract, but the code entry is bumped by all the users of all the pool contracts, so extending the lifetime of the code separately (by less) would make up for a better distribution of the fees across the network.

Impacted Components

Only WASM contract executables are impacted by this as SAC's don't have any binary associated on the ledger and their logic is hardcoded in the environment.

Implementation

The implementation is fairly simple, and the current implementation is based on some assumptions, mainly that:

- the user will still want to update the code and instance together, just with the possibility of specifying different thresholds and time to live.
- If bumping the code separately we also always want to accept a separate threshold (i.e if bumping instance and code separately, besides providing two ttls you are also providing two thresholds).

Thus the `extend_contract_instance_and_code_ttl_from_contract_id` function (not part of the `VmCaller` trait) would be modified to accept a threshold and ttl for the code entry as optional u32s. If the options wrap u32s then the threshold/ttl for the code are those specified, else they default to the ones specified for the instance:

```
None
...
ContractExecutable::Wasm(wasm_hash) => {
        let extend_to = extend_code_to.unwrap_or(extend_instance_to);
        let threshold = code_threshold.unwrap_or(instance_threshold);
...
```

The env.json has been modified as follows, but it could also be wise to keep the current exports as is and add new exports for these two host functions:

```
None
{
          "export": "a",
          "name": "extend_current_contract_instance_and_code_ttl_separately",
          "args": [
            {
              "name": "instance_threshold",
              "type": "U32Val"
            },
            {
              "name": "code_threshold",
              "type": "U32Val"
            },
            {
              "name": "extend_instance_to",
              "type": "U32Val"
            },
            {
              "name": "extend_code_to",
              "type": "U32Val"
            }
          ],
          "return": "Void",
          "docs": "Same as extend_current_contract_instance_and_code_ttl but with
the ability to extend code and instance with different lifetimes and thresholds."
        },
        {
          "export": "b",
          "name": "extend_contract_instance_and_code_ttl_separately",
          "args": [
            {
              "name": "contract",
              "type": "AddressObject"
            },
            {
              "name": "instance_threshold",
              "type": "U32Val"
            },
            {
              "name": "code_threshold",
              "type": "U32Val"
            },
            {
              "name": "extend_instance_to",
              "type": "U32Val"
            },
            {
              "name": "extend_code_to",
              "type": "U32Val"
            }
          ],
          "return": "Void",
          "docs": "Same as extend_contract_instance_and_code_ttl but with the
ability to extend code and instance with different lifetimes and thresholds."
        },
```