PACMON

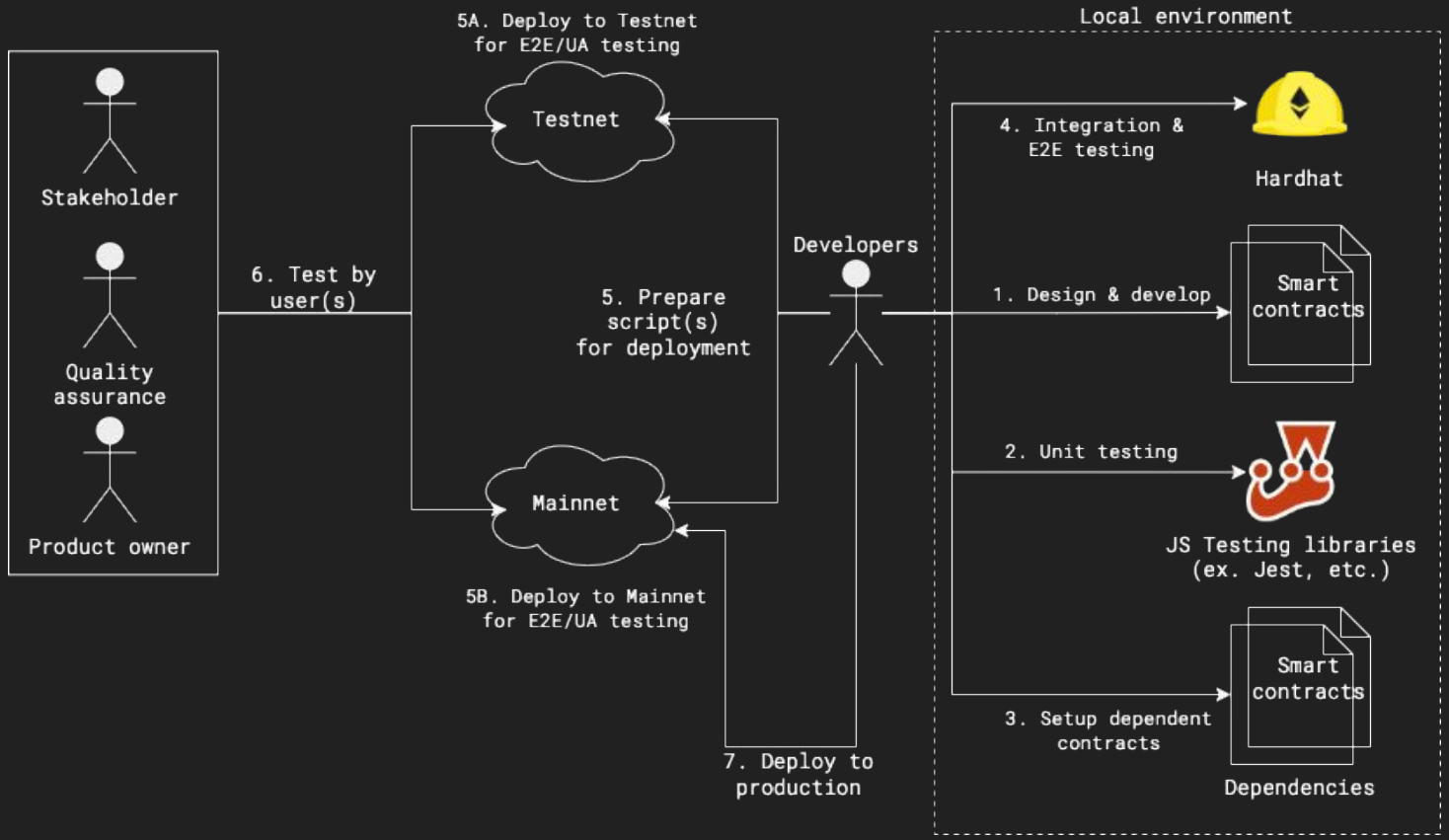
Pitch Deck

Background & Problems

Problem Statement

- Conventional end-to-end (E2E) testing approaches for Ethereum Virtual Machine (EVM) projects on testnets or mainnets are *inconvenient*
  - Testnets: Sometimes lack the required dependencies to run the projects, leading to increased overhead during preparation
  - Mainnets: can be expensive, can't manipulate state on real contracts and the blockchain state cannot be easily restored
- The testing process usually also requires the *help of Solidity engineers* to assist with environment setup and scenario execution

Conventional Development & Testing



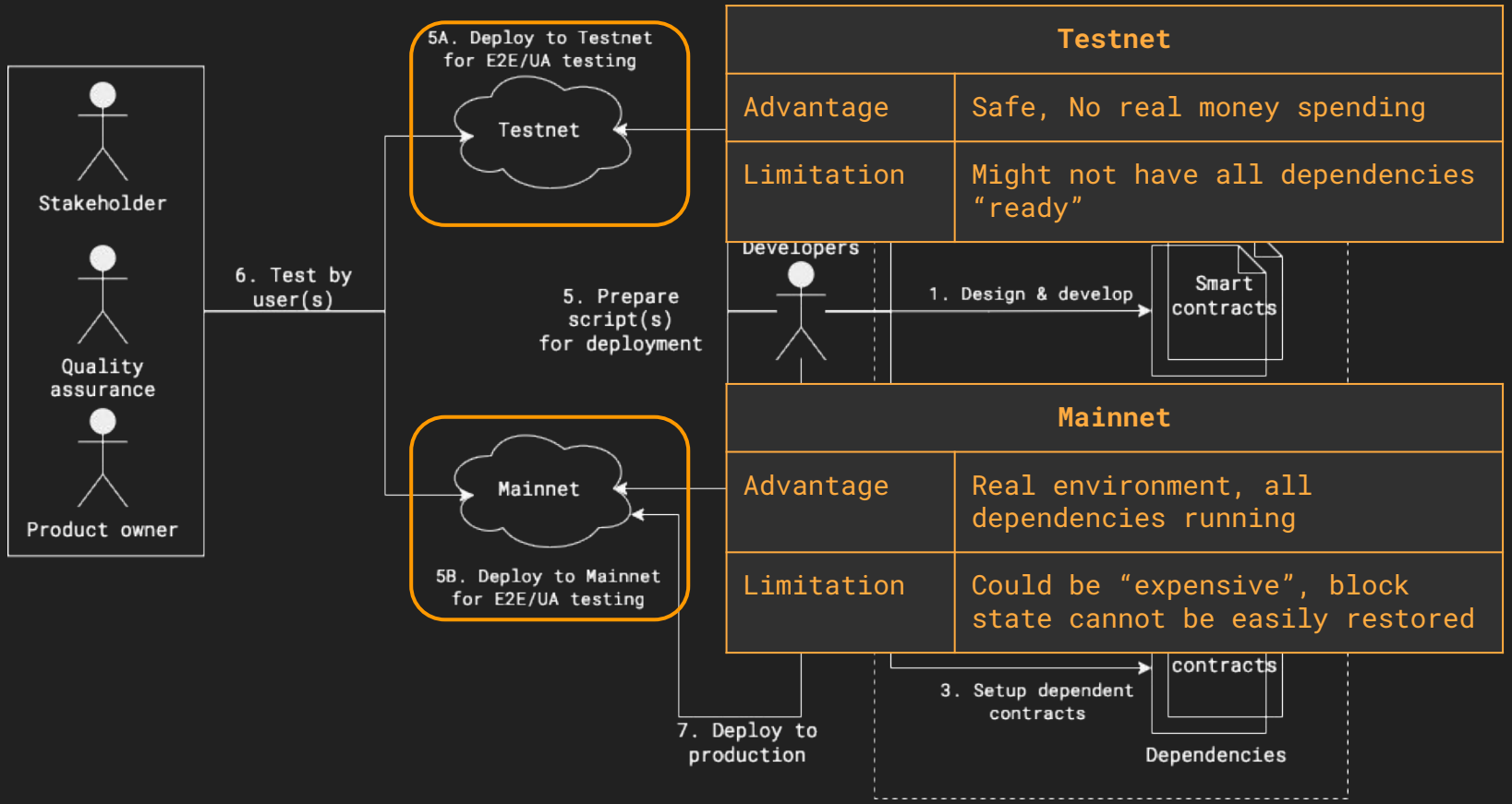
However …

We found 2 main problems from the development and testing flow

Inconvenience for testing on Testnet and Mainnet

Overhead time to support users for testing

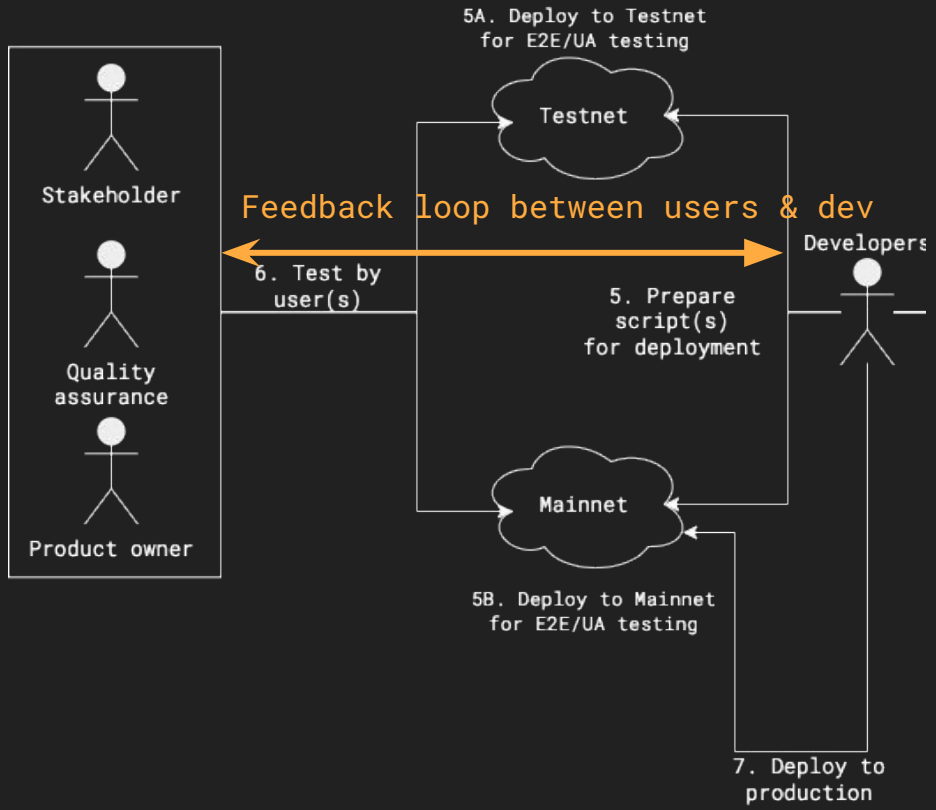
Inconvenience for testing on Testnet and Mainnet



Testnet	
Advantage	Safe, No real money spending
Limitation	Might not have all dependencies “ready”

Mainnet	
Advantage	Real environment, all dependencies running
Limitation	Could be “expensive”, block state cannot be easily restored

Overhead time to support users for testing



Example actions that users need developer's help
Can you help adjust the token's price to XX?
Can you help mint XX more tokens?
Can you help restore the blockchain state?
More …

Result: Overhead on developers to support requests

Dependencies

Goal for Our Proposal Solution

- A tool that could help making E2E testing *more convenience* and *save engineer's assistance effort*

And we introduce you to …

PACMON 

Solution: PACMON

PACMON Features

We broke down core features from the 2 problems we're interested. Let's divide our users into: *developers* and *testers* (ex. QA, PO, stakeholders)

Inconvenience for testing on Testnet and Mainnet

Overhead time to support users for testing

Let's dig deeper into user stories in the next slide

Inconvenience for testing on Testnet and Mainnet

As a developer, I want to deploy or redeploy smart contracts on testing environment with ease

With the help from concepts of templates, plugins, projects and nodes (next slide)

As a developer, I want to have a full control over the deployed testing environment

Developer gets RPC url from spawned nodes and they can do anything with nodes

Overhead time to support users for testing

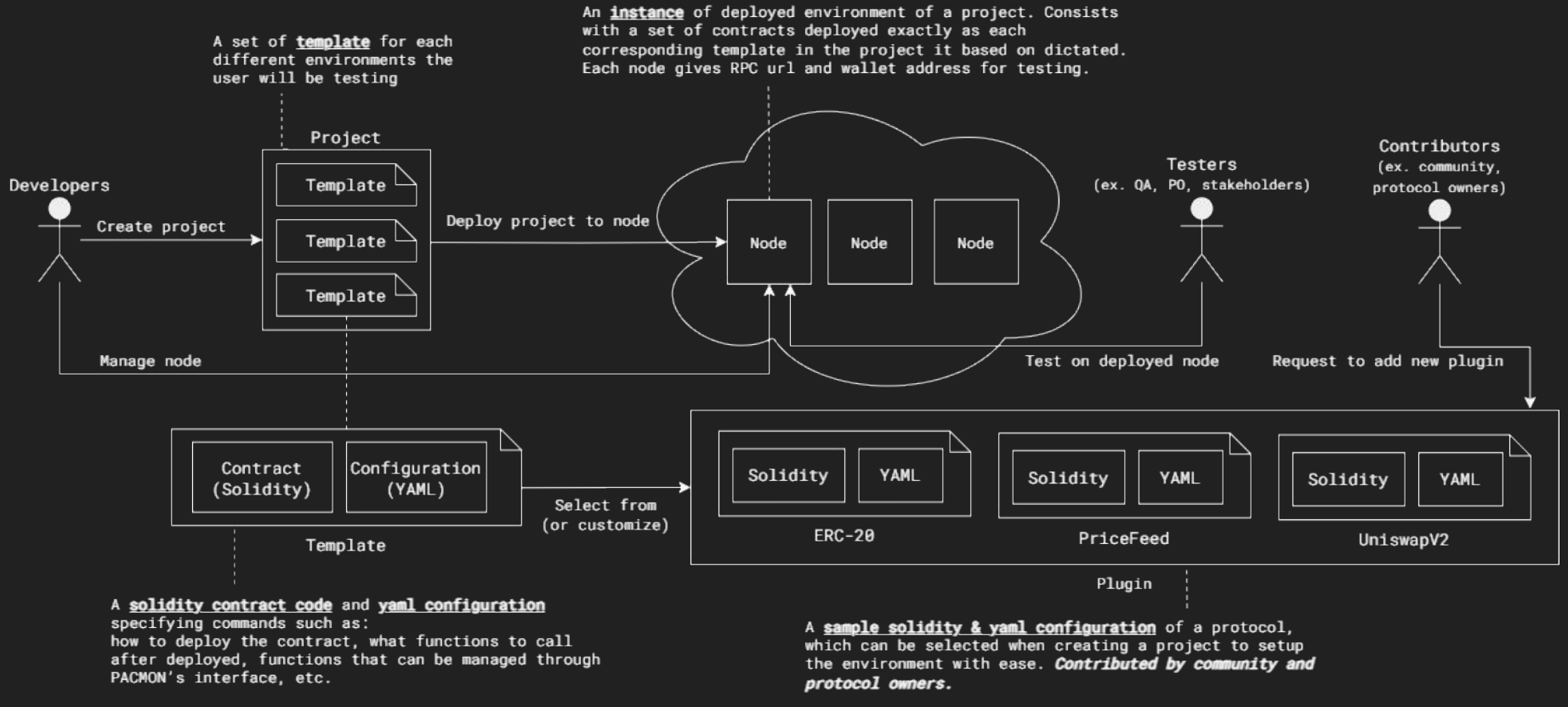
As a tester, I want to do E2E testing on testing environment

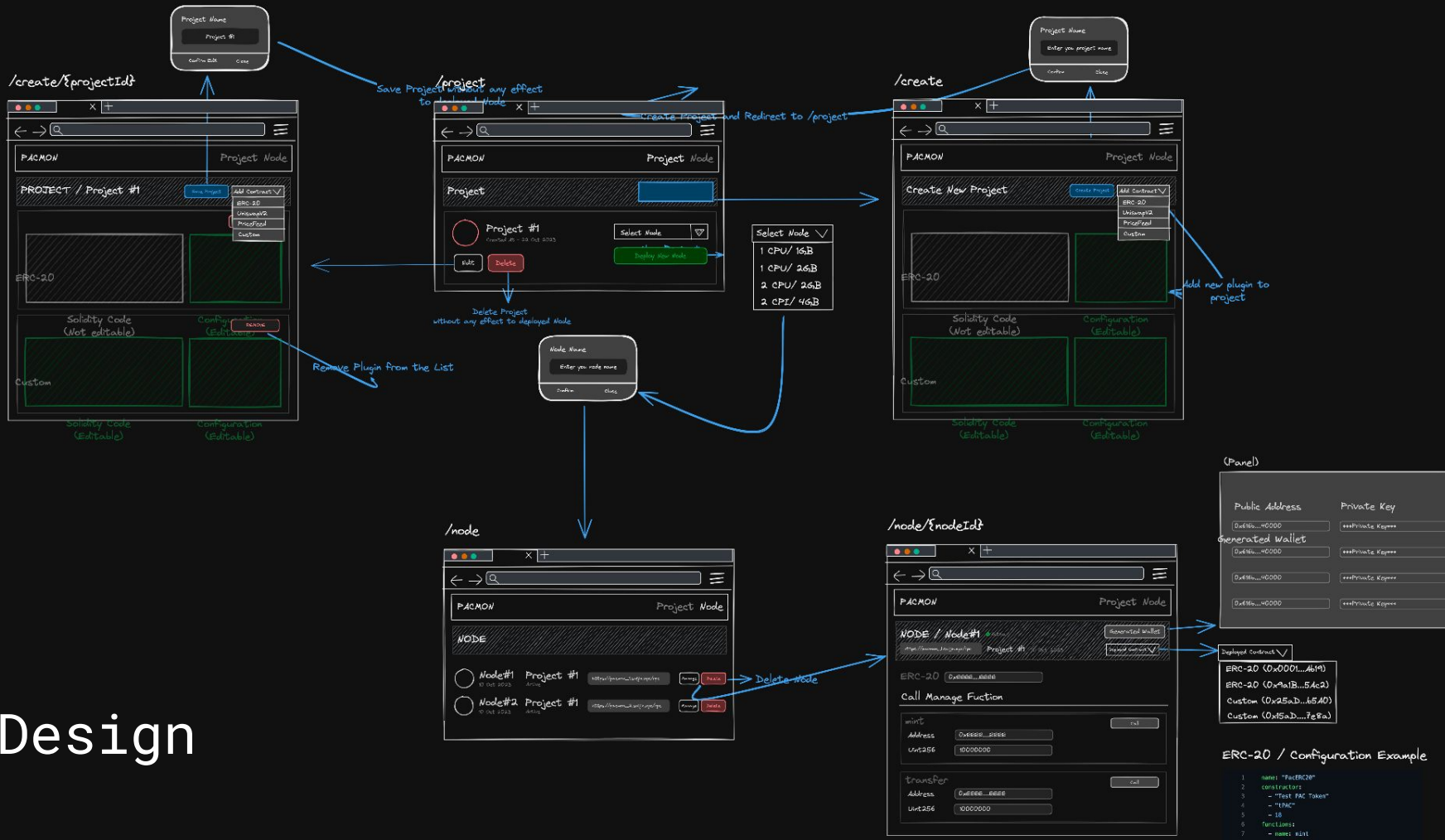
Once tester gets RPC url from developer, they can access it anywhere and do the testing

As a tester, I want to execute certain actions on testing environment without requiring effort from developers

Manage node page allows tester to do some action according to the configuration of smart contracts

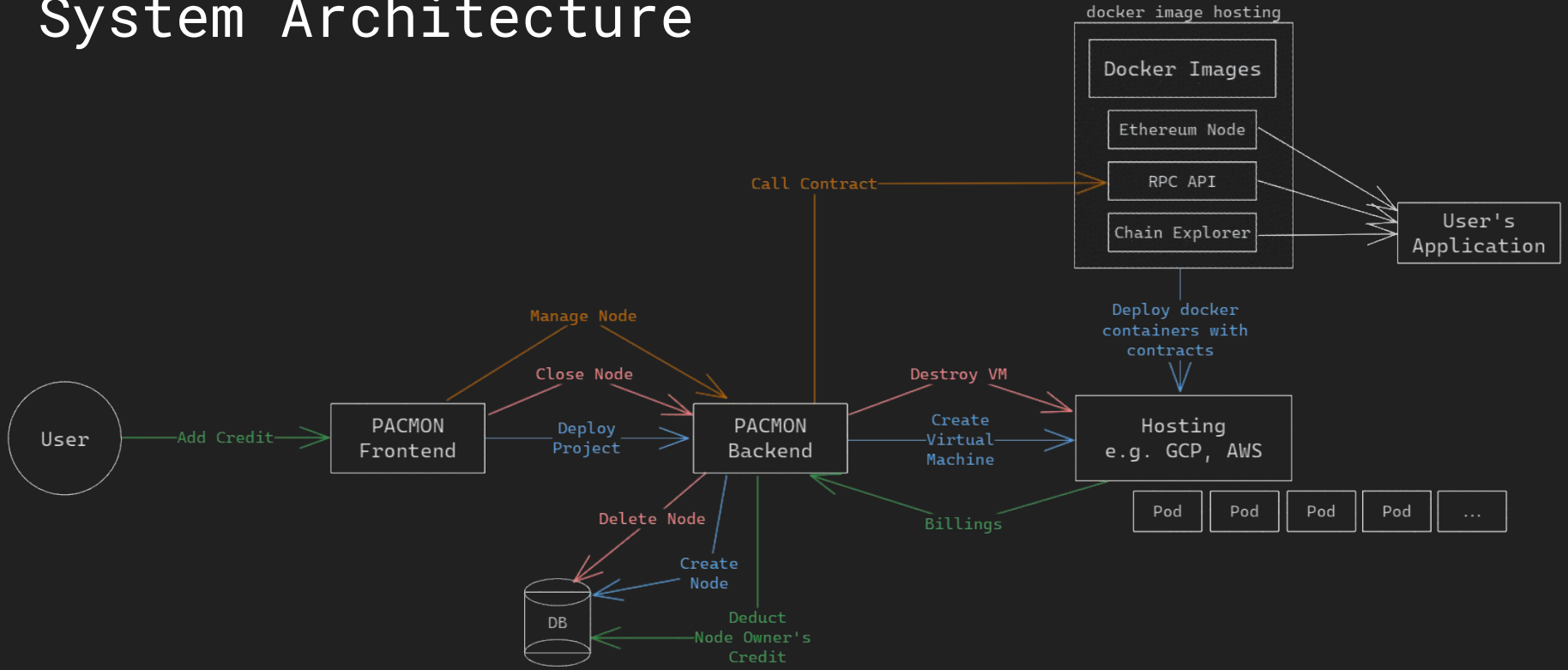
Core Concepts




/create/{projectId}
PACMON
Project Node
PROJECT / Project #1
Add Contract
ERC-20
UniswapV2
PriceFeed
Custom
ERC-20
Solidity Code
(Not editable)
Configuration
(Editable)
README
Custom
Solidity Code
(Editable)
Configuration
(Editable)
Project Name
Project #1
Confirm
Close
Save Project without any effect
to deployed Node
/project
PACMON
Project Node
Project
Project #1
Select Node
Edit
Delete
Deploy New Node
Delete Project
without any effect to deployed Node
Select Node
1 CPU/ 1GB
1 CPU/ 2GB
2 CPU/ 2GB
2 CPU/ 4GB
Node Name
Enter you node name
Confirm
Close
Remove Plugin from the List
/create
Project Name
Enter you project name
Confirm
Close
Create Project and Redirect to /project
PACMON
Project Node
Create New Project
Create Project
Add Contract
ERC-20
UniswapV2
PriceFeed
Custom
ERC-20
Solidity Code
(Not editable)
Configuration
(Editable)
Custom
Solidity Code
(Editable)
Configuration
(Editable)
Add new plugin to
project
/node
PACMON
Project Node
NODE
Node#1
Project #1
Manage
Delete
Node#2
Project #1
Manage
Delete
Delete Node
/node/{nodeId}
PACMON
Project Node
NODE / Node#1
Generated Wallet
Project #1
Deployed Contract
ERC-20
Call Manage Fuction
mint
Address
Uint256
Call
transfer
Address
Uint256
Call
(Panel)
Public Address
Private Key
Generated Wallet
Deployed Contract
ERC-20 (0x0001...4b19)
ERC-20 (0x9a1B...54c2)
Custom (0x25aD...b5A0)
Custom (0x15aD....7e8a)
ERC-20 / Configuration Example
name: "PacERC20"
constructor:
- "Test PAC Token"
- "tPAC"
- 18
functions:
- name: mint
arguments:
- "{{ADMIN}}"

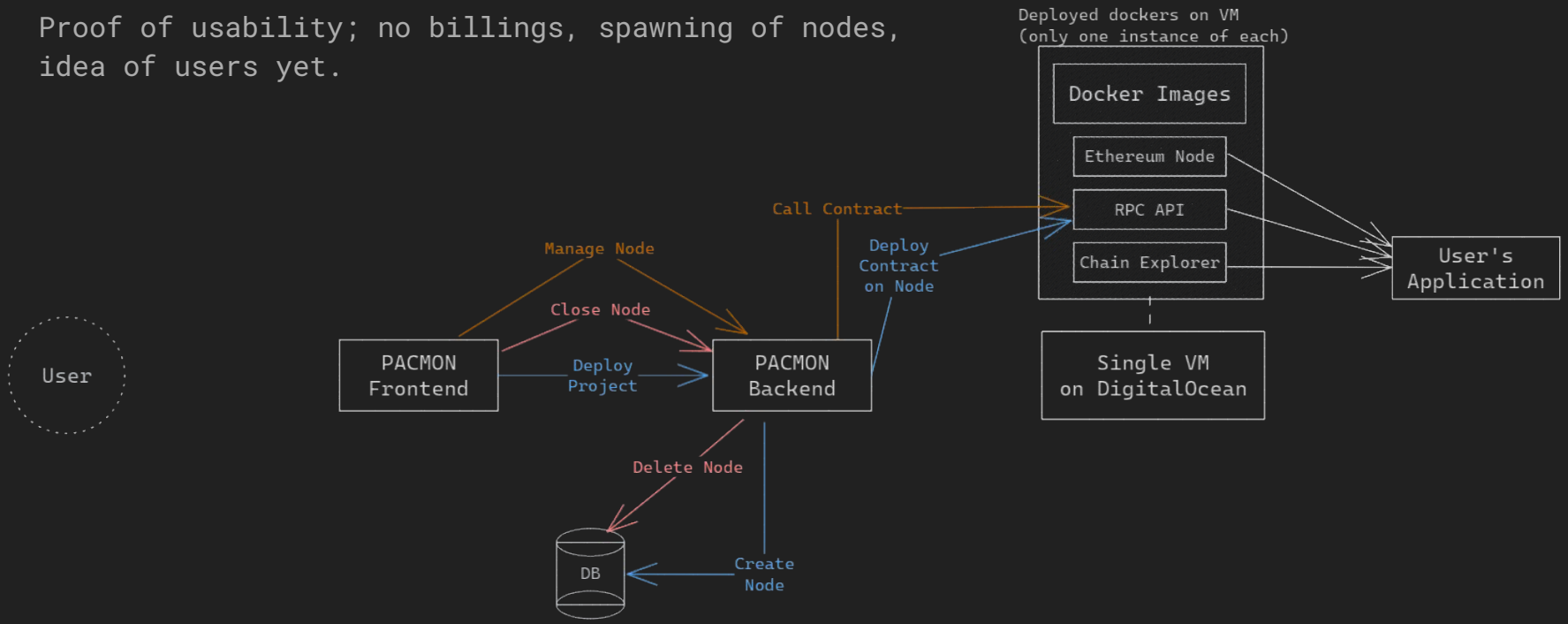
Design

System Architecture

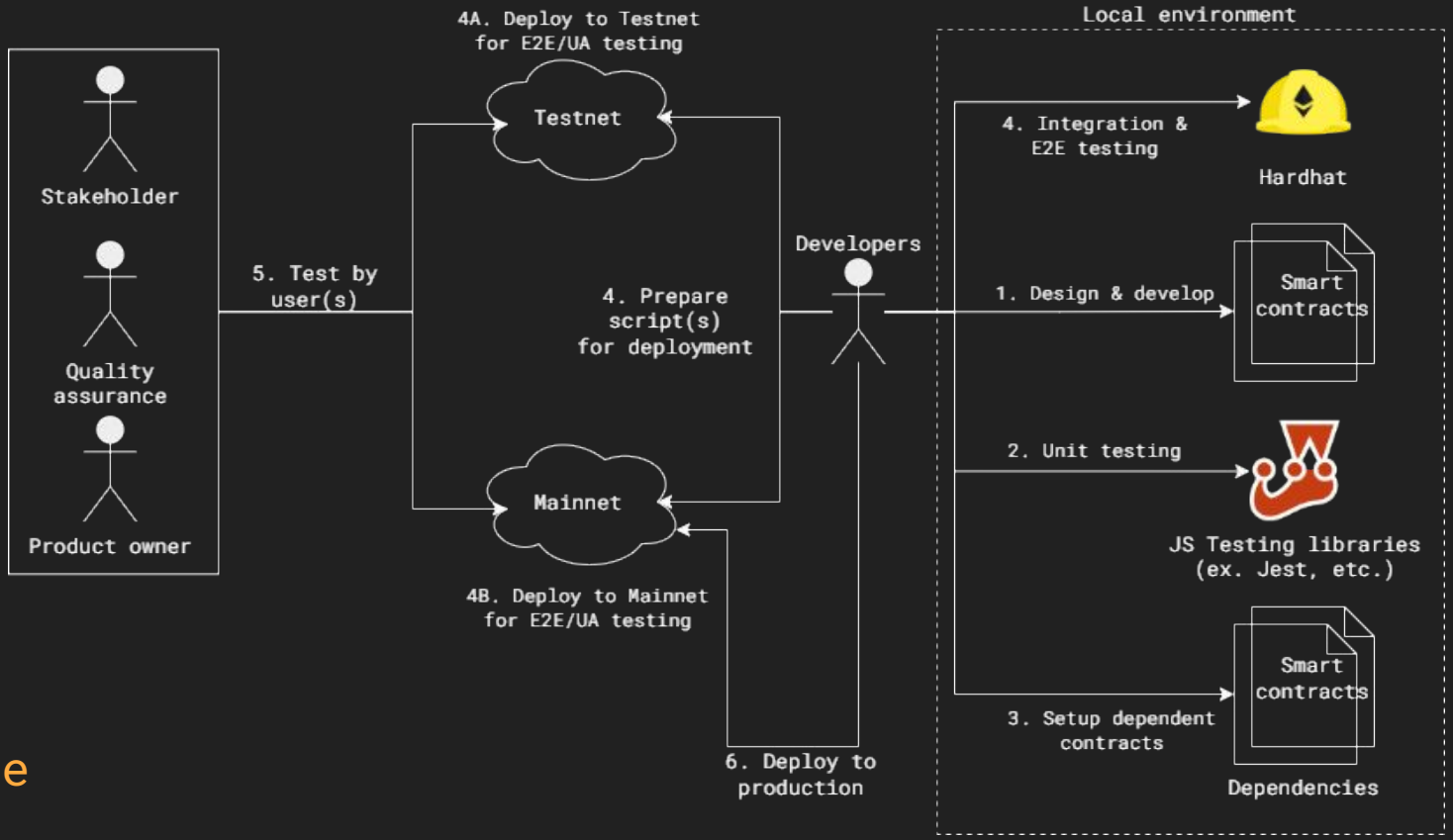


System Architecture (Proof-of-Concept)

Proof of usability; no billings, spawning of nodes, idea of users yet.

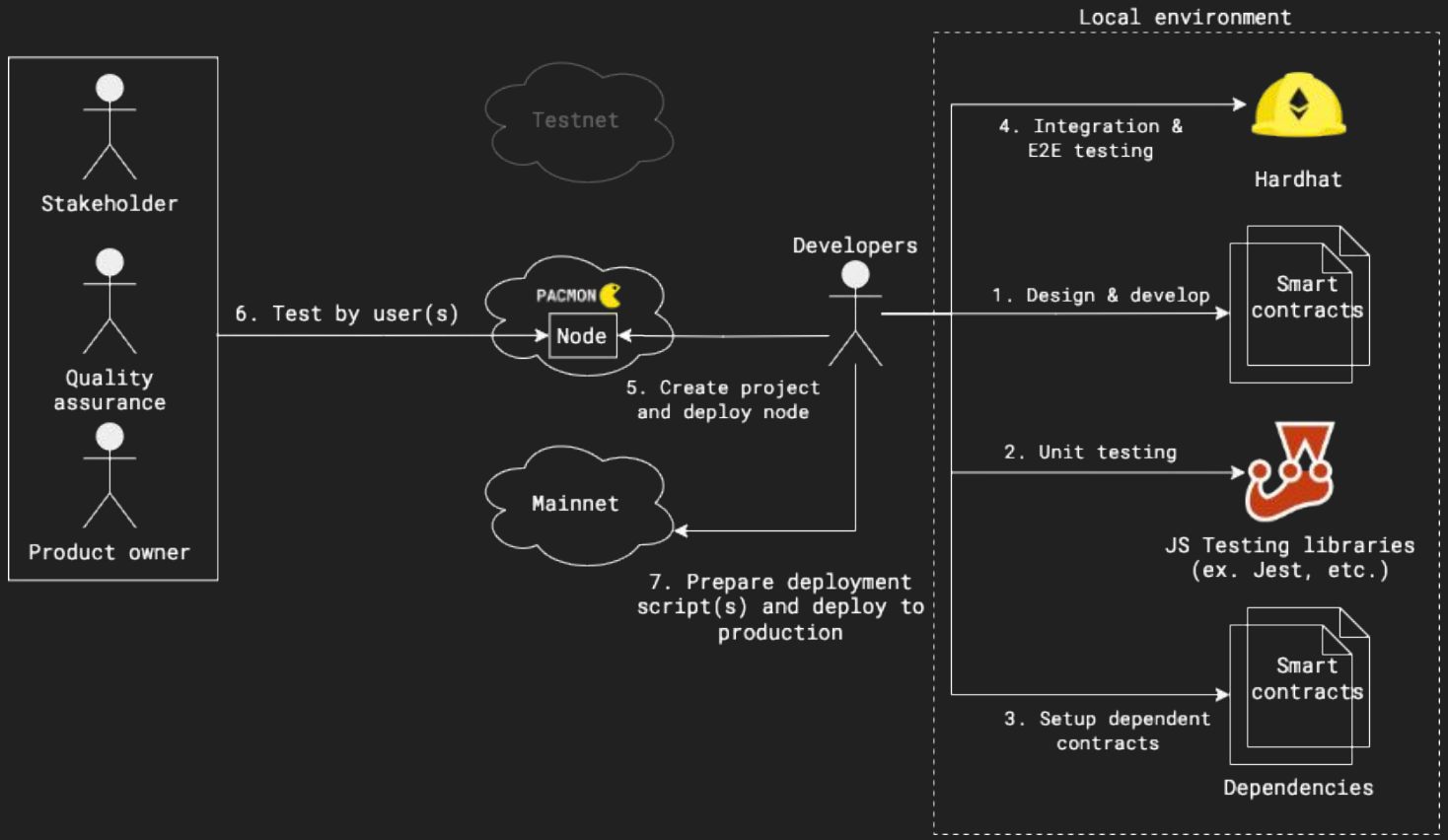


How PACMON fits into development & testing?



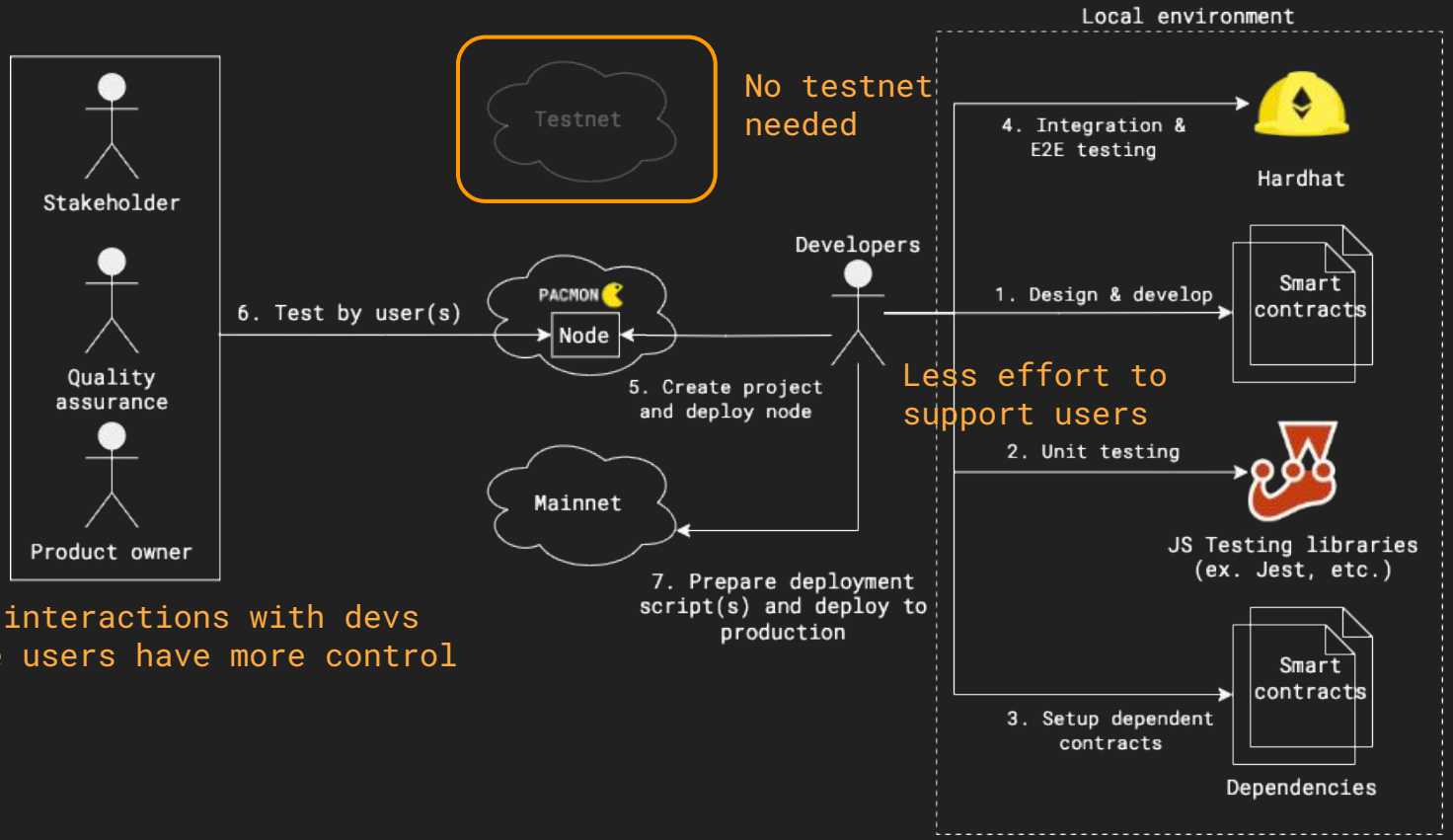
Before

How PACMON fits into development & testing?



After

How PACMON fits into development & testing?



Less interactions with devs since users have more control

After

Demo

- Production url: https://pacmon.suijin.xyz/
- Limitations
  - No separation of projects/nodes for each users
  - Support manual e2e testing only
  - Deleting node doesn't truly destroy the Ethereum node, though it will disappear from nodes list
  - Template configurations are limited inside the scope of examples we provided
  - *Only supports solidity v0.8.19*
- Full demo video: TBD
- Repositories:
  - PAC-interface: https://github.com/PacmonTeam/pac-interface
  - PAC-backend: https://github.com/PacmonTeam/pac-backend
  - PAC-plugins: Open for contribution at https://github.com/PacmonTeam/pac-plugins

Roadmap

Market Position

WEB2.0

WEB3.0

Monetization

Here are our 3 possible ideas for the product to earn:

Subscription

- Tier based features
- Subscription per seat

Pay as you go

- Pay per usage depending on node spec or feature

Enterprise plan

- Enterprise local network installation
- Support team

Product Roadmap

Q1	Q2	Q3	Q4	Future
Organization	Deployment hook in Github/Gitlab	Build and deploy on the fly	Automatic recommended server spec	AI audit services
Spawn Node on different scale	Blockchain indexer micro service	Local network installation (enterprise plan)	Self-hosted open source to introduce the tool	Scale to non-evm chains (ex. Solana, Cosmos, Sui, Aptos, etc.)
Security (authentication)	Automated script (eg. oracle price update, liquidate bot)	Automatic e2e testing		

Meet the PACTEAM

Our small team consists of 3 enthusiastic life strugglers

Chin

Progress along the way is more vital than the destination

Do: Product & Frontend

Analytically creative.
Also, he daydreams a lot.

Do: Backend

Aik

Pun

Self proclaimed introvert, but his previous manager disapproved

Do: Frontend