Timelock Wallet Hands-on Lab

This hands-on, step-by-step guide walks you through building, deploying, initializing, and testing the Timelock Wallet contract using Rust, Stylus, and Foundry's `cast`.

---

0. Prerequisites

- Rust toolchain (`rustup`)
- Node.js & npm (for foundry install, optional)
- Foundry (`curl -L https://foundry.paradigm.xyz | bash`)
- Docker (for local Stylus devnet)

---

1. Clone the Timelock Repo

```
cd stylus-timelock
```

---

2. Install Rust, WASM target, and cargo-stylus

```
rustup update
rustup target add wasm32-unknown-unknown
cargo install --force cargo-stylus
```

---

3. Review the code

Open `src/lib.rs` in your editor. The contract should have an `init()` function, not a constructor.

---

4. Build and check

```
cargo stylus check
```

---

5. (Optional) Run a Local Stylus Node

```
git clone https://github.com/OffchainLabs/nitro-devnode.git
cd nitro-devnode# Wait for readiness; your RPC is now http://localh
```

\

---

6. Deploy the Contract

```
ccargo stylus deploy \
  --endpoint='http://localhost:8547' \
  --private-
key="0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659" \
  --estimate-gas# Note the contract address from output (e.g. 0x4a2ba922...)
```

---

7. Initialize the Timelock

Pick an unlock timestamp (e.g. now+60s):

```
date +%s  # e.g. 1719876543
```

```
cast send --rpc-url 'http://localhost:8547' \
  --private-key
0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659 \
  0x[contract-address] "init(uint256)" 1719876543
# Wait for the transaction receipt.
```

---

8. Query contract state

```
cast call --rpc-url 'http://localhost:8547' 0x[contract-address] "owner()
(address)"
cast call --rpc-url 'http://localhost:8547' 0x[contract-address] "unlock_time()
(uint256)"
```

---

9. Deposit ETH

```
cast send --rpc-url 'http://localhost:8547' \
  --private-key
0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659 \
  0x[contract-address] "deposit()" --value 0.01ether
```

---

10. Withdraw (test locked/unlocked)

Try before unlock time (should revert):

```
cast send --rpc-url 'http://localhost:8547' \
  --private-key
0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659 \
  0x[contract-address] "withdraw(address)" 0x[your-address]
```

After unlock time:

```
cast send --rpc-url 'http://localhost:8547' \
  --private-key
0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659 \
  0x[contract-address] "withdraw(address)" 0x[your-address]
```

---

11. (Bonus) Extend the lock

```
cast send --rpc-url 'http://localhost:8547' \
  --private-key
0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659 \
  0x[contract-address] "extend_lock(uint256)" 1729999999
cast call --rpc-url 'http://localhost:8547' 0x[contract-address] "unlock_time()
(uint256)"
```

---

12. (Bonus) Test error handling

Try calling init a second time (should revert)

```
cast send --rpc-url 'http://localhost:8547' \
  --private-key
0xb6b15c8cb491557369f3c7d2c287b053eb229daa9c22138887752191c9520659 \
  0x[contract-address] "init(uint256)" 1729999999
```

---

13. Troubleshooting

- execution reverted: not initialized? Did you call init?
- Owner zero? You didn't call init yet.
- Funds not transferring? Unlock time not reached.
- Invalid address? Check for typos/0x prefix.

---

14. (Optional) Clean up

If running docker node:

```
docker stop [container_id]
```

---

15. Next steps

- Try deploying to Sepolia: set --endpoint to a Stylus Sepolia endpoint.
- Write a UI using viem or ethers.js.

Shell Script Template

```
DEPLOYER_KEY=0xb6b15...
cargo stylus deploy --private-key $DEPLOYER_KEY | tee deploy.log
ADDR=$(grep 'Contract deployed at' deploy.log | awk '{print $NF}')
UNLOCK=$(date +%s)
cast send --rpc-url 'http://localhost:8547' --private-key $DEPLOYER_KEY $ADDR
"init(uint256)" $UNLOCK
cast call --rpc-url 'http://localhost:8547' $ADDR "owner()(address)"
cast call --rpc-url 'http://localhost:8547' $ADDR "unlock_time()(uint256)"
```