Medium Search

Complete Guide: Installing Foundry on Windows with WSL

4 min read · May 25, 2025

PMartin Follow

Listen Share

Follow me on LinkedIn for more blockchain development content.

What is Foundry?

Foundry represents the next generation of Ethereum development tools. Built entirely in Rust, it offers unprecedented speed and efficiency compared to traditional JavaScript-based frameworks like Hardhat or Truffle.

The toolkit consists of four powerful components:

→ **Forge** — A lightning-fast testing framework
→ **Cast** — Swiss Army knife for Ethereum RPC interactions
→ **Anvil** — Local Ethereum node for seamless development
→ **Chisel** — Interactive Solidity REPL

Why Choose WSL for Windows Development?

Windows Subsystem for Linux (WSL) bridges the gap between Windows convenience and Linux performance. For blockchain developers, this means accessing the robust Unix toolchain while maintaining your familiar Windows workflow.

The benefits are substantial: native performance, full package manager access, and seamless integration with your existing Windows setup.

Prerequisites Checklist

Before diving in, ensure you have:

• Windows 10 (version 2004+) or Windows 11
• Administrator privileges
• Basic command-line familiarity
• Stable internet connection

Phase 1: WSL Installation and Setup

Installing WSL

Open PowerShell as Administrator and execute:

```
wsl --install
```

This single command handles everything: enabling WSL features, installing the kernel, setting WSL 2 as default, and downloading Ubuntu.

> **Important:** Restart your computer after installation completes.

Ubuntu Configuration

Launch Ubuntu from the Start menu and complete the initial setup:

1. Create your username and password

2. Update the system packages:

```
sudo apt update && sudo apt upgrade -y
```

3. Install essential development tools:

```
sudo apt install -y curl git build-essential
```

Phase 2: Rust Installation

Since Foundry is built with Rust, we need it as a prerequisite:

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

Select the default installation when prompted, then refresh your environment:

```
source ~/.bashrc
```

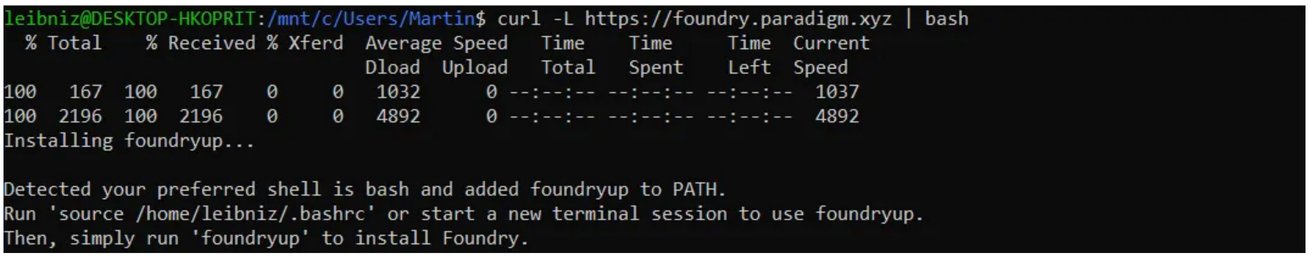
Verify the installation:

```
rustc --version
cargo --version
```

Phase 3: Foundry Installation

Using the Official Installer

The `foundryup` installer is the recommended approach:



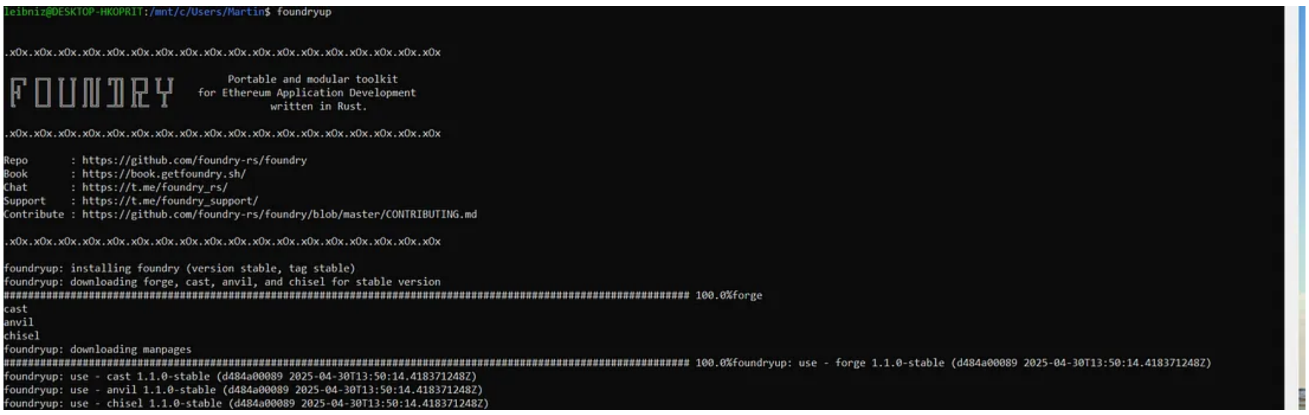
```
leibniz@DESKTOP-HKOPRIT:/mnt/c/Users/Martin$ curl -L https://foundry.paradigm.xyz | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   167  100   167    0     0   1032      0 --:--:-- --:--:-- --:--:--  1037
100  2196  100  2196    0     0   4892      0 --:--:-- --:--:-- --:--:--  4892
Installing foundryup...

Detected your preferred shell is bash and added foundryup to PATH.
Run 'source /home/leibniz/.bashrc' or start a new terminal session to use foundryup.
Then, simply run 'foundryup' to install Foundry.
```

```
curl -L https://foundry.paradigm.xyz | bash
```

Reload your terminal environment:

```
source ~/.bashrc
```

Complete the installation:



```
leibniz@DESKTOP-HKOPRIT:/mnt/c/Users/Martin$ foundryup

.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x

 FOUNDRY          Portable and modular toolkit
           for Ethereum Application Development
                       written in Rust.

.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x

Repo       : https://github.com/foundry-rs/foundry
Book       : https://book.getfoundry.sh/
Chat       : https://t.me/foundry_rs/
Support    : https://t.me/foundry_support/
Contribute : https://github.com/foundry-rs/foundry/blob/master/CONTRIBUTING.md

.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x.x0x

foundryup: installing foundry (version stable, tag stable)
foundryup: downloading forge, cast, anvil, and chisel for stable version
######################################################################################################## 100.0%forge
cast
anvil
chisel
foundryup: downloading manpages
######################################################################################################## 100.0%foundryup: use - forge 1.1.0-stable (d484a00089 2025-04-30T13:50:14.418371248Z)
foundryup: use - cast 1.1.0-stable (d484a00089 2025-04-30T13:50:14.418371248Z)
foundryup: use - anvil 1.1.0-stable (d484a00089 2025-04-30T13:50:14.418371248Z)
foundryup: use - chisel 1.1.0-stable (d484a00089 2025-04-30T13:50:14.418371248Z)
```

```
foundryup
```

Verification

Confirm all tools are properly installed:

```
forge --version
```

```
cast --version
anvil --version
chisel --version
```

Each command should return version information.

Phase 4: Creating Your First Project

Project Initialization

Create and initialize a new Foundry project:

```
mkdir blockchain-project
cd blockchain-project
forge init
```

This generates a complete project structure:

```
blockchain-project/
├── foundry.toml          # Project configuration
├── lib/                  # External dependencies
├── script/               # Deployment scripts
├── src/                  # Smart contract source
│   └── Counter.sol       # Example contract
└── test/                 # Test suite
    └── Counter.t.sol     # Example tests
```

Exploring the Example Contract

Examine the generated counter contract:

```
cat src/Counter.sol
```

This simple contract demonstrates fundamental Solidity patterns and serves as your starting point.

Phase 5: Building and Testing

Compilation

Compile your smart contracts:

```
forge build
```

This creates the `out/` directory containing compiled bytecode and ABI files.

Running Tests

Execute your test suite:

```
forge test
```

For detailed output including gas usage:

```
forge test -vv
```

Coverage Analysis

Generate test coverage reports:

```
forge coverage
```

Phase 6: Local Development with Anvil

Starting Your Local Blockchain

Launch Anvil in a dedicated terminal:

```
anvil
```

Anvil provides 10 pre-funded accounts with 10,000 ETH each, perfect for development and testing.

Contract Deployment

Deploy your contract to the local network:

```
forge script script/Counter.s.sol \
  --rpc-url http://localhost:8545 \
  --private-key 0xac0974bec39a17e36ba4a6b4d238ff944bacb478cbed5efcae784d7bf4
  --broadcast
```

> **Note:** This private key is from Anvil's default accounts — safe for local development only.

Phase 7: Advanced Workflows

Dependency Management

Install popular libraries like OpenZeppelin:

```
forge install OpenZeppelin/openzeppelin-contracts
```

Configuration Optimization

Enhance your `foundry.toml`:

```
[profile.default]
src = "src"
out = "out"
libs = ["lib"]
solc_version = "0.8.19"
optimizer = true
optimizer_runs = 200
remappings = ["@openzeppelin/=lib/openzeppelin-contracts/"]
```

```
[rpc_endpoints]
mainnet = "https://eth-mainnet.alchemyapi.io/v2/YOUR_API_KEY"
sepolia = "https://eth-sepolia.g.alchemy.com/v2/YOUR_API_KEY"
```

Blockchain Interaction with Cast

Interact with your deployed contracts:

```
# Check block number
cast block-number --rpc-url http://localhost:8545
```

```
# Query account balance
cast balance 0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266 --rpc-url
http://localhost:8545

# Call contract function
cast call YOUR_CONTRACT_ADDRESS "number()" --rpc-url http://
localhost:8545
```

Troubleshooting Common Issues

WSL Won't Start: Verify virtualization is enabled in BIOS settings and all Windows features are properly installed.

Command Not Found Errors: Ensure you've sourced your bashrc file after installation: `source ~/.bashrc`

Compilation Failures: Check that your Solidity version in `foundry.toml` matches your contract requirements.

Network Connection Problems: Confirm Anvil is running and accessible on the specified port (default: 8545).

Security Best Practices

1. Never commit private keys to version control
2. Use environment variables for sensitive data
3. Implement comprehensive test coverage for all contract functions
4. Regular dependency updates to avoid vulnerabilities
5. Gas optimization using Foundry's built-in profiling tools

Next Steps and Resources

With your Foundry environment set up, you're ready to build sophisticated smart contracts. Consider exploring:

• Advanced testing patterns with fuzz testing
• Integration with front-end frameworks
• Multi-chain deployment strategies
• Continuous integration with GitHub Actions

Essential Resources:

→ Official Foundry Book
→ GitHub Repository
→ Community Discord

Conclusion

Foundry paired with WSL creates a powerful development environment that combines Windows familiarity with Linux performance. This setup positions you at the forefront of smart contract development, leveraging cutting-edge tools that prioritize speed, reliability, and developer experience.

The blockchain development landscape is evolving rapidly, and Foundry represents the future of how we build, test, and deploy smart contracts. With this guide, you're equipped to join the next generation of blockchain developers.

Solidity Foundry Windows Crypto Web3

Follow

Written by PMartin

15 followers · 0 following

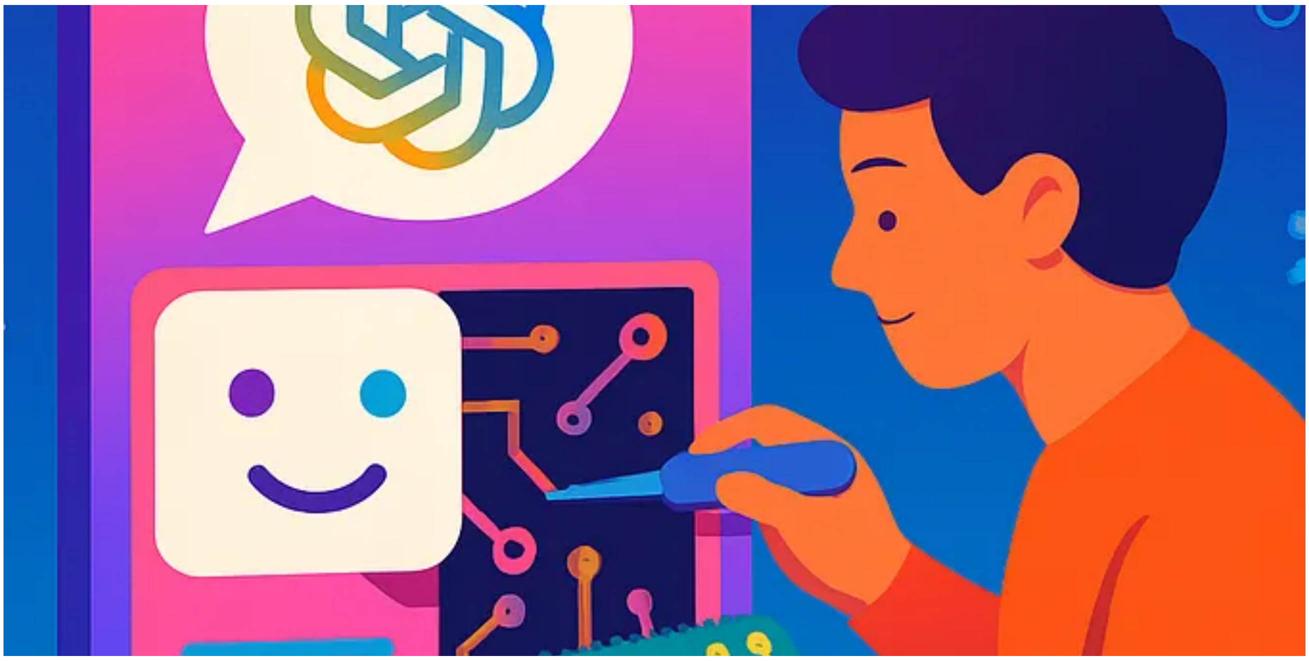
Software Engineer

No responses yet

Write a response

What are your thoughts?

More from PMartin



In JavaScript in Plain English by PMartin

I Reverse-Engineered ChatGPT's UI. Here's What OpenAI Doesn't Want You to Know

How a 3-week deep dive into ChatGPT's interface revealed the billion-dollar psychology tricks that make AI feel "smarter"

Aug 1 52 1

In CoinsBench by PMartin

⚡ Flash Loan Mastery: From Zero to DeFi Arbitrage Hero in 3 Steps

Follow me on LinkedIn for more blockchain development content.

Jun 4 121 1

In CoinsBench by PMartin

Smart Contract Security Wars: The Ultimate Slither vs Mythril Battle

In JavaScript in Plain English by PMartin

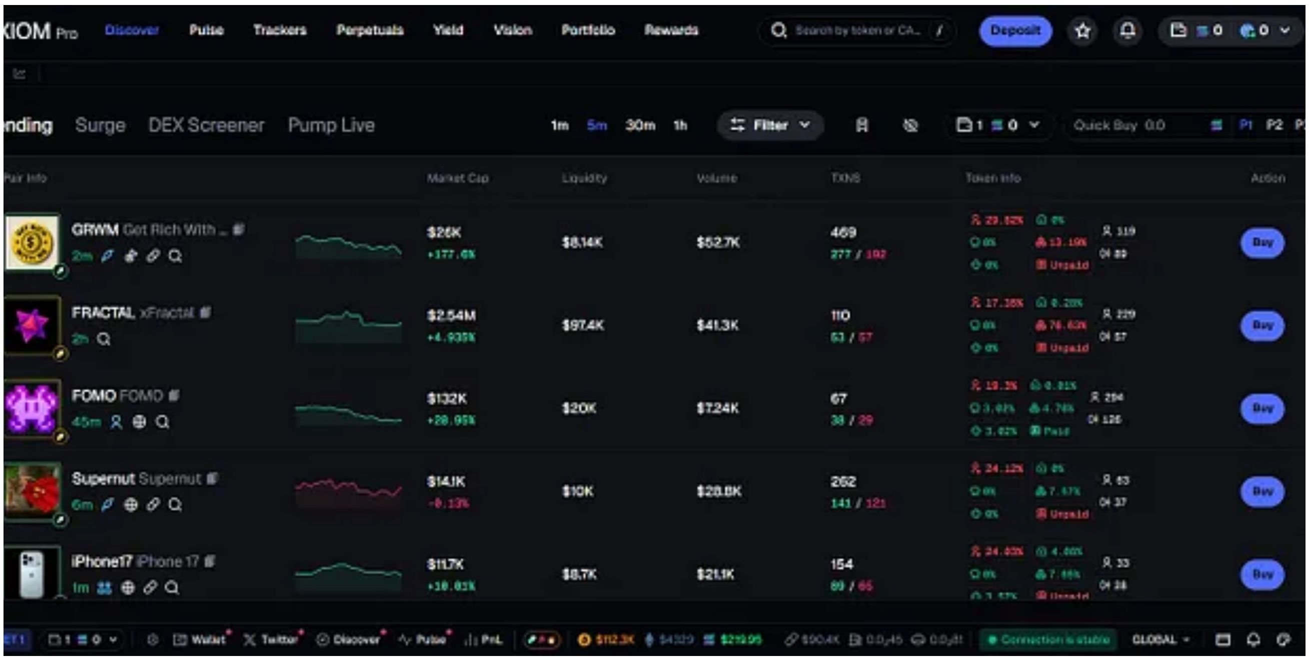
Why Web3 UX Still Sucks (And How to Fix It)

Follow me on LinkedIn for more development content.

Jun 30

See all from PMartin

Recommended from Medium



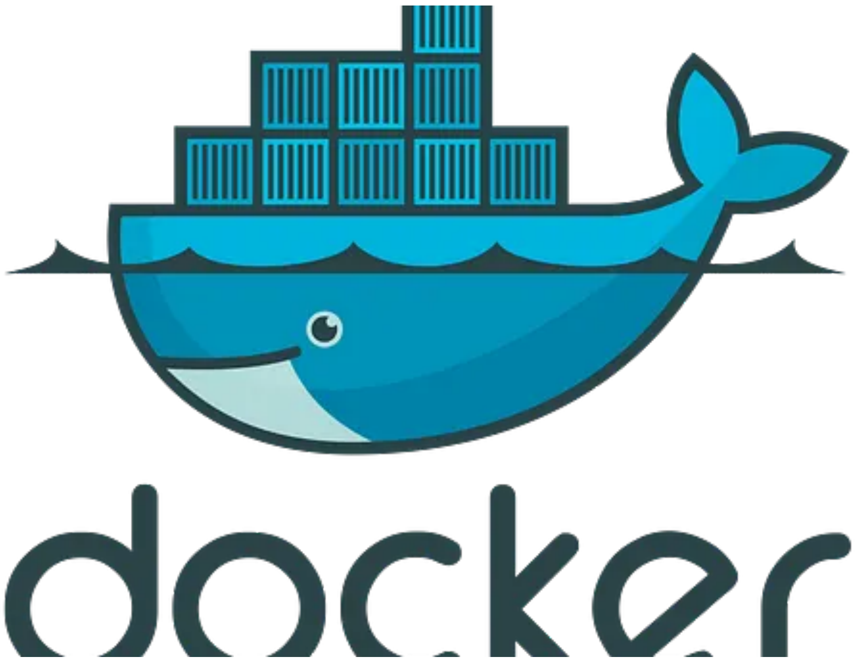
Pair Info	Market Cap	Liquidity	Volume	TXNS	Token Info	Action
GRWM Get Rich With ... 2m	$26K +177.6%	$8.14K	$52.7K	469 277 / 192	29.82% 0% 0% 13.19% 0% Unpaid 119 89	Buy
FRACTAL xFractal 2h	$2.54M +4.935%	$97.4K	$41.3K	110 53 / 57	17.38% 0.20% 0% 76.62% 0% Unpaid 229 57	Buy
FOMO FOMO 45m	$132K +28.95%	$20K	$7.24K	67 38 / 29	18.3% 0.81% 3.01% 4.76% 3.02% Paid 294 126	Buy
Supernut Supernut 6m	$14.1K -0.13%	$10K	$28.8K	262 141 / 121	24.12% 0% 0% 7.67% 0% Unpaid 63 37	Buy
iPhone17 iPhone 17 1m	$11.7K +18.01%	$8.7K	$21.1K	154 89 / 65	24.03% 4.00% 0% 7.66% 3.57% Unpaid 33 28	Buy

Dinakar

This company reached $100M in revenue in just under 4 months

It's not Lovable, Cluely, or Cursor we're talking about

Sep 10



Abhinav

Docker Is Dead — And It's About Time

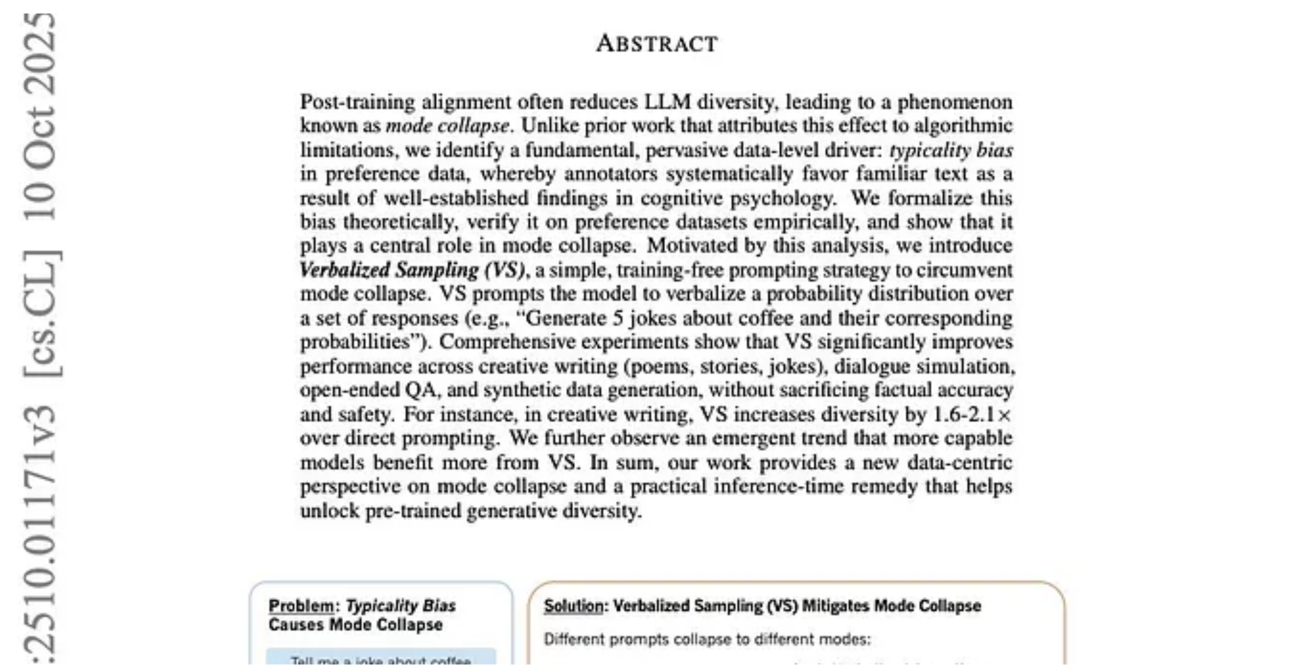
7.1K 202

In Artificial Intelligence in Plain English by Klippa

The 10 Best OCR API Providers You Need to Know in 2025

Discover the top 10 OCR APIs of 2025 and see why Klippa DocHorizon stands out from the rest!

Jul 16 254 1



:2510.01171v3 [cs.CL] 10 Oct 2025

Abstract

Post-training alignment often reduces LLM diversity, leading to a phenomenon known as *mode collapse*. Unlike prior work that attributes this effect to algorithmic limitations, we identify a fundamental, pervasive data-level driver: *typicality bias* in preference data, whereby annotators systematically favor familiar text as a result of well-established findings in cognitive psychology. We formalize this bias theoretically, verify it on preference datasets empirically, and show that it plays a central role in mode collapse. Motivated by this analysis, we introduce ***Verbalized Sampling (VS)***, a simple, training-free prompting strategy to circumvent mode collapse. VS prompts the model to verbalize a probability distribution over a set of responses (e.g., "Generate 5 jokes about coffee and their corresponding probabilities"). Comprehensive experiments show that VS significantly improves performance across creative writing (poems, stories, jokes), dialogue simulation, open-ended QA, and synthetic data generation, without sacrificing factual accuracy and safety. For instance, in creative writing, VS increases diversity by 1.6-2.1× over direct prompting. We further observe an emergent trend that more capable models benefit more from VS. In sum, our work provides a new data-centric perspective on mode collapse and a practical inference-time remedy that helps unlock pre-trained generative diversity.

Problem: *Typicality Bias* Causes Mode Collapse

Tell me a joke about coffee

Solution: Verbalized Sampling (VS) Mitigates Mode Collapse

Different prompts collapse to different modes:

In Generative AI by Adham Khaled

Stanford Just Killed Prompt Engineering With 8 Words (And I Can't Believe It Worked)

ChatGPT keeps giving you the same boring response? This new technique unlocks 2× more creativity from ANY AI model — no training required...

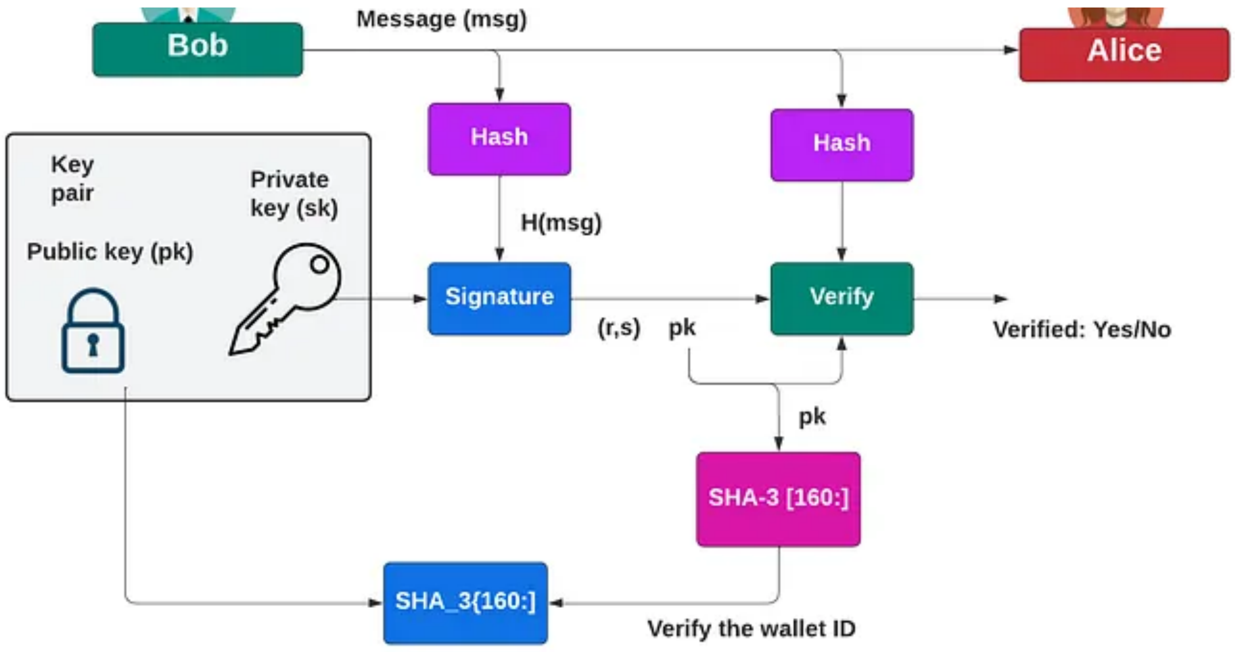
11.1K 241

Aditya Bhuyan

Bridging Elasticsearch and PostgreSQL: A Deep Dive into Integration Challenges

Introduction

Jul 15 7



In ASecuritySite: When Bob Met Alice by Prof Bill Buchanan OBE FRSE

The Post Quantum Migration of Blockchain

And, so, within the next five years, something is happening that will completely change our blockchain infrastructures — quantum robust...

Aug 2 227 9

See more recommendations