INSTRUCTIONS TO RUN THE NEAR BLOCKCHAIN PROJECT

Project Overview

This document provides step-by-step instructions for setting up and running the NEAR blockchain project using the nearcore repository. The nearcore repository contains the reference implementation of the NEAR Protocol, a scalable and developer-friendly blockchain platform.

Prerequisites

Before running the project, ensure you have the following tools installed:

1. Rust toolchain (cargo, rustc)
2. WSL (Windows Subsystem for Linux) - Ubuntu 20.04 or later
3. Git
4. Docker (optional, for containerized deployment)
5. Node.js (optional, for frontend components)

Installation Steps

1. Clone the Repository

```
cd /mnt/c/Users/USER/Documents/blockchainotp
git clone https://github.com/near/nearcore.git
```

2. Install Rust Dependencies

```
# Install Rust toolchain if not already installed
sudo apt update
sudo apt install -y cargo rustc

# Add necessary components
rustup component add clippy rustfmt
```

3. Install Additional Tools

```
# Install cargo tools for development and testing
cargo install cargo-audit cargo-deny cargo-fuzz
cargo install cargo-crev cargo-sbom
cargo install kani-verifier
```

```
# Install binaryen for WASM optimization
sudo apt install -y binaryen
```

Building the Project

1. Navigate to the Project Directory

```
cd /mnt/c/Users/USER/Documents/blockchainotp/nearcore
```

2. Build the NEAR Node

```
# Build the neard binary
cargo build -p neard --release
```

3. Initialize the Node

```
# Initialize the node with default configuration
./target/release/neard init --chain-id localnet --account-id node0
```

Running the Node

1. Start the Node

```
# Run the node
./target/release/neard run
```

2. Run in Development Mode

```
# For development and testing, you can use the sandbox
cargo install near-sandbox
near-sandbox --home /tmp/near-sandbox init
near-sandbox --home /tmp/near-sandbox run
```

Testing the Project

1. Run Unit Tests

```
# Run all unit tests
cargo test
```

2. Run Integration Tests

```
# Run integration tests
cargo test --test integration_tests
```

3. Security Auditing

```
# Check for known vulnerabilities
cargo audit

# Check for licensing issues
cargo deny check advisories
cargo deny check licenses
```

4. Code Quality Checks

```
# Run clippy for linting
cargo clippy --all-targets --all-features

# Format code
cargo fmt --all -- --check
```

5. Fuzz Testing

```
# Run fuzz tests (if configured)
cargo fuzz run fuzz_target_1
```

Working with Smart Contracts

1. Create a New Contract Project

```
# Create a new Rust project for smart contracts
cargo new --lib my_contract
cd my_contract
```

2. Configure Cargo.toml

Add the following to your Cargo.toml:

```
[lib]
crate-type = ["cdylib", "rlib"]

[dependencies]
near-sdk = "4.1.1"

[profile.release]
codegen-units = 1
opt-level = "z"
lto = true
debug = false
panic = "abort"
overflow-checks = true
```

3. Build the Contract

```
# Build the contract for NEAR
cargo build --target wasm32-unknown-unknown --release
```

4. Optimize the Contract

```
# Optimize the WASM file
wasm-opt -Oz --strip-debug target/wasm32-unknown-
unknown/release/my_contract.wasm -o my_contract.optimized.wasm
```

Development Tools

1. NEAR CLI

Install and use the NEAR CLI for interacting with the network:

```
# Install NEAR CLI
npm install -g near-cli

# Login to NEAR
near login

# Deploy a contract
near deploy --wasmFile my_contract.optimized.wasm --accountId your-
account.testnet
```

```
# Call a contract method
near call your-account.testnet my_method '{"args": "values"}' --accountId your-
account.testnet
```

2. Workspaces-rs for Testing

For integration testing with workspaces-rs, add this to your Cargo.toml:

```
[dev-dependencies]
near-workspaces = "0.10"
tokio = { version = "1.14", features = ["full"] }
```

Example test:

```
use near_workspaces::sandbox;

#[tokio::test]
async fn test_contract() -> anyhow::Result<()> {
    let worker = sandbox().await?;
    let contract = worker.dev_deploy(&contract_wasm).await?;
    // Test contract functionality
    Ok(())
}
```

Monitoring and Observability

1. Enable Logging

```
# Run with verbose logging
./target/release/neard run --verbose
```

2. Prometheus Metrics

The node exposes metrics on port 3030 by default. You can configure Prometheus to scrape these metrics.

Troubleshooting

Common Issues and Solutions

1. **Build Failures**:
    - Ensure you have the latest Rust toolchain: `rustup update`
    - Clear cargo cache: `cargo clean`

2. **Permission Issues**:
    - Make sure you have write permissions to the project directory
    - Run commands with appropriate user privileges
3. **Network Issues**:
    - Check your internet connection
    - Ensure firewall settings allow network communication
4. **Dependency Issues**:
    - Update dependencies: `cargo update`
    - Check for version conflicts in Cargo.toml

Useful Commands

```
# Check Rust version
rustc --version

# Check Cargo version
cargo --version

# List installed cargo tools
cargo install --list

# Check node status
./target/release/neard view-state
```

Next Steps

1. Explore the nearcore documentation: https://nomicon.io/
2. Learn NEAR smart contract development: https://docs.near.org/
3. Join the NEAR developer community: https://near.org/
4. Contribute to the project by following the guidelines in CONTRIBUTING.md

Additional Resources

- NEAR Protocol Documentation: https://docs.near.org/
- NEAR SDK Documentation: https://near-sdk.io/
- NEAR Examples: https://examples.near.org/
- NEAR GitHub: https://github.com/near

This document was created to help you set up and run the NEAR blockchain project. For any issues or questions, refer to the official documentation or community resources.