This is a **Forgejo Actions workflow** (Codeberg's CI/CD system, highly compatible with GitHub Actions syntax) for a **Rust-based Solana program developed with the Anchor framework**.

It runs on pushes to the main branch and on pull requests, performing two main tasks:

1. Build and test the program.
2. If the push is to main, deploy it to Solana **devnet**.

The runner is codeberg-medium, a hosted medium-sized runner provided by Codeberg for more resource-intensive jobs (e.g., compilation and tests that need extra RAM/CPU).

Overall Structure

YAML

```
name: Rust + Anchor CI

on:
  push:
    branches: [ main ]
  pull_request:

jobs:
  build-and-test:     # Always runs (on PRs and pushes)
    ...

  deploy-devnet:      # Only runs on direct pushes to main (not
```

PRs)

```
    needs: build-and-
test
    if: github.ref ==
'refs/heads/main' &&
github.event_name ==
'push'
    ...
```

Note: Even though it's Codeberg, it still uses ${{ github.ref }} and ${{ github.event_name }} — this is normal in Forgejo Actions due to compatibility.

Job 1: build-and-test

This job checks out the code, sets up the environment, builds the program, runs tests, and prepares program IDs. Key steps explained:

- **Checkout** — Standard action to clone the repository.
- **Cache Cargo registry + git + target** — Speeds up subsequent runs by caching Rust dependencies and build artifacts. Key based on OS and Cargo.lock hash.
- **Install dependencies (including entropy for Solana)** Installs haveged, a daemon that generates additional entropy using CPU timing variations. **Why?** In headless CI environments (no mouse/keyboard/disk activity), entropy can be very low, causing Solana

keypair generation or cryptographic operations to block or slow down dramatically.

- **Install rust toolchain** — Uses a custom action to install stable Rust with rustfmt and clippy.
- **Install Solana CLI (latest stable, non-interactive)** Uses a custom installer script (https://solana-install.solana.workers.dev) that downloads and sets up the latest Solana tools without modifying the PATH permanently. The workflow manually adds it to PATH.
- **Install Anchor CLI (latest)** Installs avm (Anchor Version Manager) from the Coral-XYZ repo, then installs and switches to the latest Anchor version.
- **Clean broken program ID from anchor toml** Removes any existing [programs.*] sections and program_id lines from Anchor.toml. This prevents CI from using stale/hardcoded program IDs that might conflict.
- **Synch Anchor Keys** Runs anchor keys sync → generates new program keypairs if needed and updates Anchor.toml with fresh program IDs. Then configures git (as "CI bot") and commits/pushes the updated Anchor.toml back to the repository with message "CI: Sync program ids [ci skip]" (the [ci skip] prevents triggering

another CI run).

- **Verify versions** — Prints versions of Rust, Cargo, Solana, and Anchor for debugging.
- **Debug secret** — Prints length and first/last 20 chars of the base64-encoded secret (helps confirm the secret is correctly passed without leaking the full value).
- **Load devnet keypair & sync** Decodes the DEVNET_DEPLOY_KEY_BASE64 secret (a base64-encoded Solana keypair file) and saves it as ~/.config/solana/id.json. Configures Solana CLI to use this keypair and devnet URL. Requests a small airdrop (2 SOL) for transaction fees (ignored if rate-limited). Runs anchor keys sync again (in case something changed).
- **Build and Test**anchor build → compiles the Rust program to a Solana BPF binary. anchor test → runs the full test suite (spins up a local validator, deploys, runs TypeScript/Mocha tests).

There's a commented-out section that would print the base64 of a newly generated keypair if no secret was provided — useful for initially creating the secret.

Job 2: deploy-devnet

This job runs **only after successful build-and-test** and **only on pushes to main**.

It re-sets up most of the same environment (checkout, haveged, Rust, Solana CLI, Anchor CLI), then:

- **Setup devnet wallet and deploy** Same as above: decodes the secret keypair, configures Solana CLI for devnet, airdrops a bit of SOL. Runs anchor deploy --provider.cluster devnet → builds (if needed) and deploys the program(s) to Solana devnet using the provided keypair.
- **Deployment success** Prints a success message and lists known program keys with anchor keys list.

Why this workflow is structured this way

- **Program IDs must be consistent**: Anchor programs are deployed to specific on-chain addresses derived from keypairs. To keep them stable across branches/PRs, the workflow syncs and commits updated IDs on main.
- **Secure deployment**: Deployment uses a secret-held keypair with funds, restricted only to main branch pushes (prevents PRs from deploying with your funds).
- **Entropy fix**: Critical for reliable Solana operations in CI.
- **Caching**: Makes repeated runs much faster.
- **Separate jobs**: Allows testing

on every PR without risking deployment.

This is a solid, production-like CI/CD setup for an Anchor/Solana project hosted on Codeberg. If you have specific parts you'd like more detail on (e.g., why certain commands are used), let Grok. ChatGpt or Deepseek know!