Tracking Nixpkgs Merged Pull Requests

When is the fix available? A 5-Minute Guide to Tracking Nixpkgs PRs!

Made with Marp by Leonard Sheng Sheng Lee.

The Problem

✅ Nixpkgs PR approved and merged.

🔄 Run `nix flake update`.

❌ **Updates not available yet.**

Typical Deployment Paths

Fast Lane (< 500 Rebuilds)

`master` ➡️ `<nix-channel>`

Slow Lane (1000+ Rebuilds)

`staging` 🔁 `staging-next` ↩️

↪️ `master` ➡️ `<nix-channel>`

Why the Slow Lane?

Mass rebuilds (Go, Python, GCC)

5000+ packages affected.

Days of build time.

Risk of breakages.

Solution: Batch in `staging` branch.

Labels Tell the Story

Check your PR labels:

- `10.rebuild-linux: 5001+` → **staging** (Slow Lane)
- `10.rebuild-linux: 1-10` → **master** (Fast Lane)
- `1.severity: security` → Backport to Stable

🏷️ **Labels =** 📌 **Pins routes on the** 🗺️ **Map.**

Example: Go 1.25.6

PR #480465 (Security Fix)

Labels:

`1.severity: security` `10.rebuild-linux: 5001+`

Path:

`staging` → (Slow Lane) → Your System

Example: Ruby Backport

PR #451386

```
COMMIT_ID=$(gh pr view 451386 \
    --repo NixOS/nixpkgs \
    --json mergeCommit \
    | jq --raw-output '.mergeCommit.oid')

gh api \
    "repos/NixOS/nixpkgs/compare/${COMMIT_ID}...nixos-unstable" \
    --jq '.status'
```

If `ahead` or `identical`, then it's ✅ available.

Quick Reference

Rebuilds		Target Branch	Speed
< 500	=	`master`	Fast
500-1000	⇒	`staging`	Medium
1000+	=	`staging`	Slow (~ Weeks)

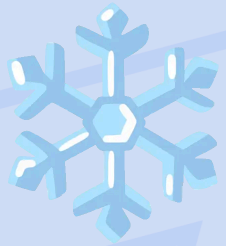
See status.nixos.org page.

Summary

1. **Merging ≠ Availability** - PRs follow different paths.
2. **Labels** - They predict timeline.
3. **Staging ≈ Patience** - Mass rebuilds take time.
4. **Track, Don't Guess** - Use Git{,Hub} commands.

Thanks!

sheeeng.github.io/slides



Made with Marp by Leonard Sheng Sheng Lee.