Nixtamal

Fulfilling input pinning for Nix—without flakes

Table of Contents

Keys features

- Automate the manual work of input pinning, allowing to lock & refresh inputs
- Declaritive KDL manifest file over imperative CLI flags
- Host, forge, VCS-agnostic
- Choose eval time fetchers (builtins) or build time fetchers (Nixpkgs, default)—which opens up fetching Darcs, Pijul, & Fossil
- Supports mirrors
- Override hash algorithm on a per-project & per-input basis—including BLAKE3 support[*]
- Custom freshness commands

- No experimental Nix features required

---

[*] Opting in requires enabling blake3-hashes experimental feature which requires Nix 2.31+.

Showcase

Set up

```
$ nixtamal set-up

Fetching fresh value for  nixpkgs  …
Prefetching input  nixpkgs  … (this may take a while)
Prefetched  nixpkgs .
Making manifest file @ version:1.0.0

$ nixtamal tweak
```

Tweak the manifest with your $EDITOR

```
version "0.4.0"
// By default in this project, use experimental BLAKE3 algorithm for
// quicker, safer hashing
default-hash-algorithm BLAKE3
// Define & even reuse patches
patches {
    // Unique name for referencing in manifest inputs
    chroma-0.22.0 "https://patch-diff.githubusercontent.com/raw/NixOS/nixpkgs/pul
        // Override the project default hash algorithm
        hash algorithm=SHA-512 expected="1mdsfx204bgia572fydnmjy78dkybbcnjx20qn9l
    }
}
// Define inputs
```

```
inputs {
    // Unique name for referencing in Nix
    nixpkgs {
        // Fetch an archive with string templating support
        archive {
            url "https://github.com/NixOS/nixpkgs/archive/{{fresh_value}}.tar.gz"
        }
        hash algorithm=SHA-256
        // Apply patches to the source now while awaiting review
        patches chroma-0.22.0
        // cURL an Atom feed for updates, stat a directory, whatever you
        // need so long as it returns a string, you can use it!
        // This also means you can prevent downloading massive files by
        // deciding yourself what “fresh” means to you.
        fresh-cmd {
            $ git ls-remote --branches "https://github.com/NixOS/nixpkgs.git" --re
            | cut -f1
        }
    }
    nixtamal {
        // Use VCSs not supported by `builtins`
        darcs {
            repository "https://darcs.toastal.in.th/nixtamal/stable"
            // fallback to mirrors when a host is down
            mirrors "https://smeder.ee/~toastal/nixtamal.darcs"
        }
        fresh-cmd {
            $ curl -sL "https://darcs.toastal.in.th/nixtamal/stable/_darcs/weak_ha
        }
    }
    // Even static JSON files can be inputs
    mozilla-tls {
        file {
            url "https://ssl-config.mozilla.org/guidelines/{{fresh_value}}.json"
            mirrors "https://raw.githubusercontent.com/mozilla/ssl-config-generat
        }
        // Scrape a webpage for the latest version
        fresh-cmd {
            $ curl -sL "https://wiki.mozilla.org/Security/Server_Side_TLS"
            | htmlq -t "table.wikitable:last-of-type > tbody > tr:nth-child(2) > 
        }
```

```
    }
    // Download & pin a model from Hugging Face, then can be used like with
    // `specialArgs = { inherit inputs; }`:
    //
    // services.llama-cpp = {
    //    enable = true;
    //    model = "${inputs.Qwen2_5-Coder-7B-Instruct}";
    // }
    Qwen2_5-Coder-7B-Instruct {
        file {
            url "https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct-GGUF/resol
        }
        fresh-cmd {
            $ curl -fsL "https://huggingface.co/api/models/Qwen/Qwen2.5-Coder-7B-
            | jq -r ".[0].id"
        }
    }
}
```

Lock or refresh your new inputs

```
$ nixtamal lock
…

$ nixtamal refresh
…
```

Using in a project

```
let
   # Import the inputs attrset from the $NIXTAMAL_DIRECTORY (default:
   # nix/tamal), which takes a configuration attrset:
   #
   # {
   #    system ? builtins.currentSystem,
   #    nixpkgs ? null,
```

```
    #    bootstrap-nixpkgs-lock-namee ? null,
    # }:
    #
    # system : string
    #    The system architecture (useful for pure evaluation)
    # bootstrap-nixpkgs : derivation
    #    A Nixpkgs source for bootstrapping. This is useful if using Nixtamal
    #    from inside another another pinning tool, OR, if you are willing to
    #    trade off a bit of purity at the fetcher level for better  performance
    #    / saving data, you can pass { bootstrap-nixpkgs = <nixpkgs>; } from the
    #    host system as the system's Nixpkgs fetchers are likely stable enough
    #    for your bootstrapping needs (& we feature check some of the API).
    # bootstrap-nixpkgs-lock-name : string
    #    Key name from inputs to use as the bootstrapping Nixpkgs if you
    #    want/need inputs.nixpkgs to point to something else.
    inputs = import ./nix/tamal { };

    pkgs = import inputs.nixpkgs {
       overlay = [
          # exposes the “dag-yo” package
          (import "${inputs.my-cool-package}/nix/overlay")
       ];
    };
in
pkgs.dag-yo # punted!!
```

Comparison

NIX PINNING TOOL COMPARISONS

Pinning tool	Nixtamal	Nix channels	Nix flakes	npins	niv	Yae
Same state per machine	yes	only if manually pinning	yes	yes	yes	yes
Per-project support	yes	no	yes	yes	yes	yes

Pinning tool	Nixtamal	Nix channels	Nix flakes	npins	niv	Yae
Versioned schemas	yes	-	no	yes	yes	no
Requires experimental Nix features	no	no	yes	no	no	no
Splits lockfile vs. manifest duties	yes (manifest.kdl)	no	yes (flake.nix)	no	no	no
Requires Nixpkgs	used for bootstrapping & some features require it, but possible with `fetch-time=eval`	no	no	no	no	no
Fetch CVS	no[2]	no	no	no	no	no
Fetch Darcs	yes	no	no	no	no	no
Fetch Fossil	yes	no	no	no	no	no
Fetch Git	yes	no	yes	yes	yes	yes
Fetch GNU Bazaar	no[1]	no	no	no	no	no
Fetch Mercurial	no[2]	no	yes	no	no	no
Fetch Pijul	yes	no	no	no	no	no
Fetch Subversion	no[2]	no	no	no	no	no
Fetch torrent	no[1]	no	no	no	no	no
Fetch URLs	yes	yes	yes	yes	yes	yes
User-written freshness logic	fresh-cmd	no	no	no	no	no
Version constraints	Gate with fresh-cmd or Jingoo templating (DIY)	no	no	Semver on some input kinds	no	Git tag predicate
Configure hash algorithm	yes, per-project + per-input & BLAKE3 support	no	no	no	no	no
Mirror support	yes, on supported kinds	no	no	no	no	no

Pinning tool	Nixtamal	Nix channels	Nix flakes	npins	niv	Yae
Patch support	yes, declarative	no	must apply manually or pull in a dependency to handle	must apply manually	must apply manually	must apply manually
Forge-specific URL schemes or semantics	no	no	yes	yes	yes, & defaults to proprietary MS GitHub	no
Freeze inputs for convenience	yes	no	no	yes	no	no
Bin license	GPL-3.0-or-later	LGPL-2.1-or-later	LGPL-2.1-or-later	EUPL-1.2	MIT	GPL-3.0
Main implementation language	OCaml	C++	C++	Rust	Haskell	Go
Manifest file format	KDL	-	Nix (special constraints)	-	-	-
Lockfile file format	JSON	-	JSON	JSON	JSON	JSON

---

[1] (1,2) No prefetcher exists

[2] (1,2,3) Bare-bones prefetchers: While these could be supported like `fetchurl` does, VCSs can & should provide structured output (currently supported prefetchers all output JSON).

Design constraints

- All inputs are named and listed up front
- Inputs are written declaratively in a manifest file
- Lockfiles are machine-written, not hand-edited configuration

- VCSs not in the builtins must be supported
- No forge-specific rules
- Users define how freshness is checked

Technical choices

- After bootstrapping with Nixpkgs, use its fetchers as builtins fetchers are (by design) feature-poor
- Uses nix-prefetch-scripts from Nixpkgs input value fetching
- Allows side-by-side use with other pinning tools during transition
- Nixtamal is dog-fooded on itself

Site made with Nix (dep management), Nickel (config), Soupault (SSG), Docutils (rST rendering), mandoc (manpage conversion), & sugilite256 (color scheme).