Gachix

A Binary Cache for Nix over Git

Ephraim Siegfried

16.01.2026

University of Basel

Outline

1. Motivation 2
2. Background 4
3. Design 15
4. Results 38

1. Motivation

Motivation

- Nix and Git share many commonalities
- Most notably, both operate on a directed acyclic graph
- Implementing Nix functionality with Git gives nice features “for free”:
  - Efficient peer-to-peer replication
  - Efficient storage

2. Background

2.1. Nix

Nix Package Manager

- Declarative and **functional package manager**

2.1. Nix

Nix Package Manager

- Declarative and **functional package manager**
- Enforces **reproducibility** in package builds:

2.1. Nix

Nix Package Manager

- Declarative and **functional package manager**
- Enforces **reproducibility** in package builds:
  - All dependendencies must be specified

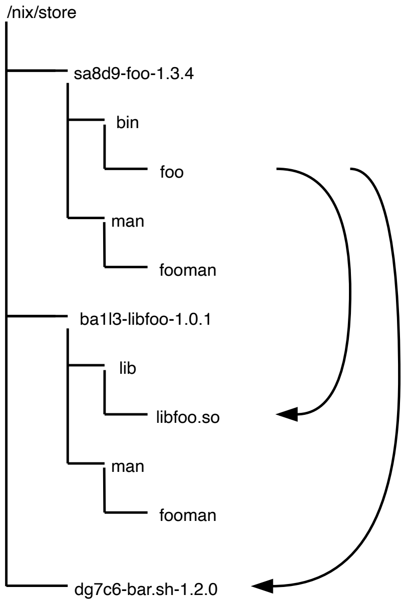
2.1. Nix

Nix Package Manager

- Declarative and **functional package manager**
- Enforces **reproducibility** in package builds:
  - All dependendencies must be specified
  - Building a Nix expression twice yields same result

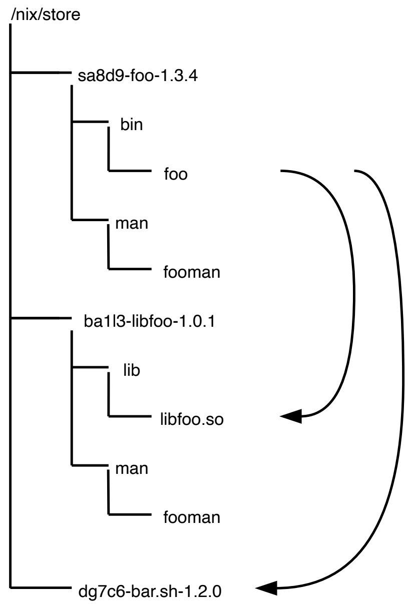
Nix Store

- **Collection of packages** and other data



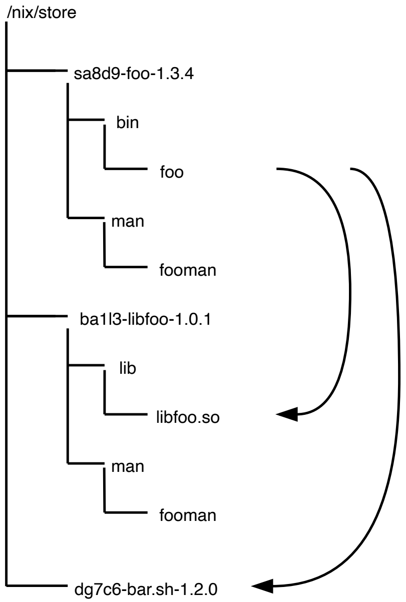
Nix Store

- **Collection of packages** and other data
- Each entry is **immutable**



Nix Store

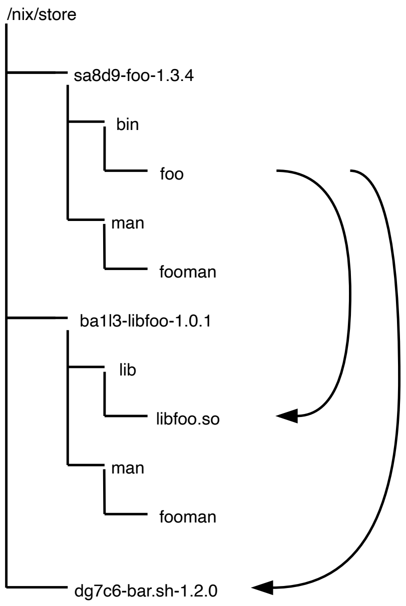
- **Collection of packages** and other data
- Each entry is **immutable**
- Uniquely **identified by hash** of dependency graph



Nix Store

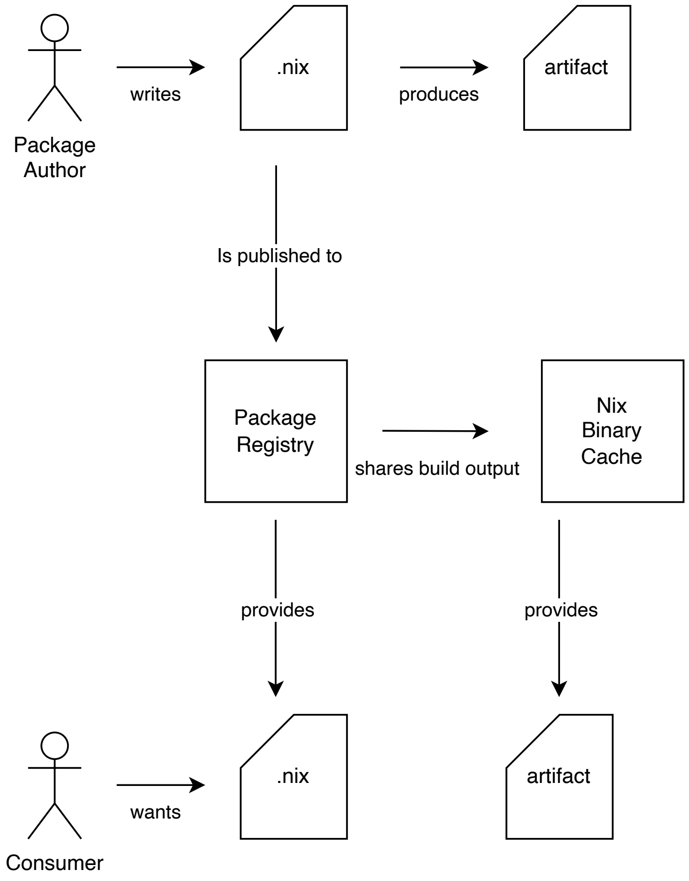
- **Collection of packages** and other data
- Each entry is **immutable**
- Uniquely **identified by hash** of dependency graph
- Entries are of the form:

```
/nix/store/<hash>-<name>-<version>
```



Deployment Pipeline

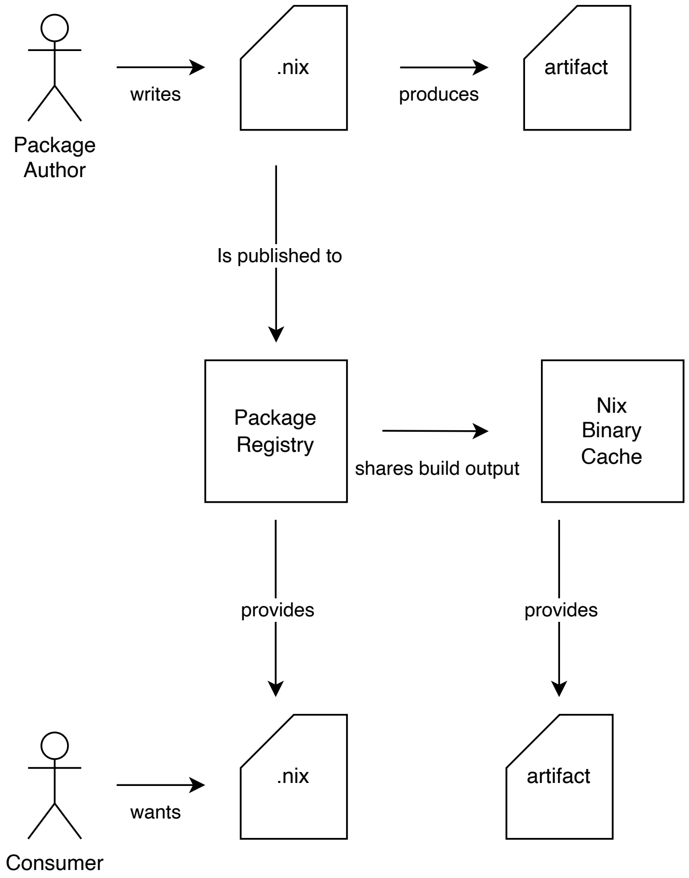
- Nix produces packages from Nix files
- A Nix file is a build recipe for a package



Deployment Pipeline

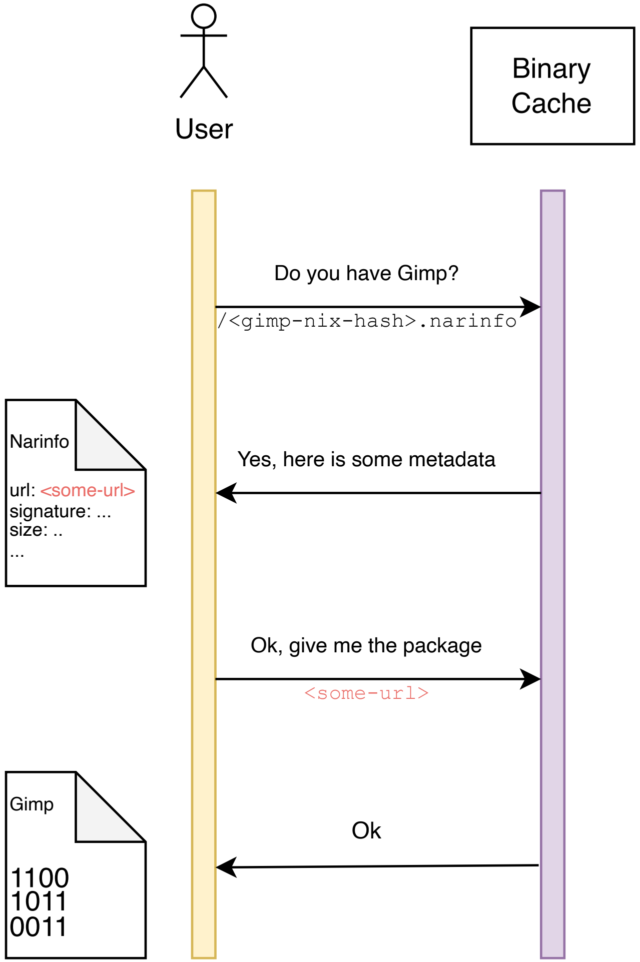
- Nix produces packages from Nix files
- A Nix file is a build recipe for a package
- Building a package can take a long time

  → Use binary caches to speed up builds



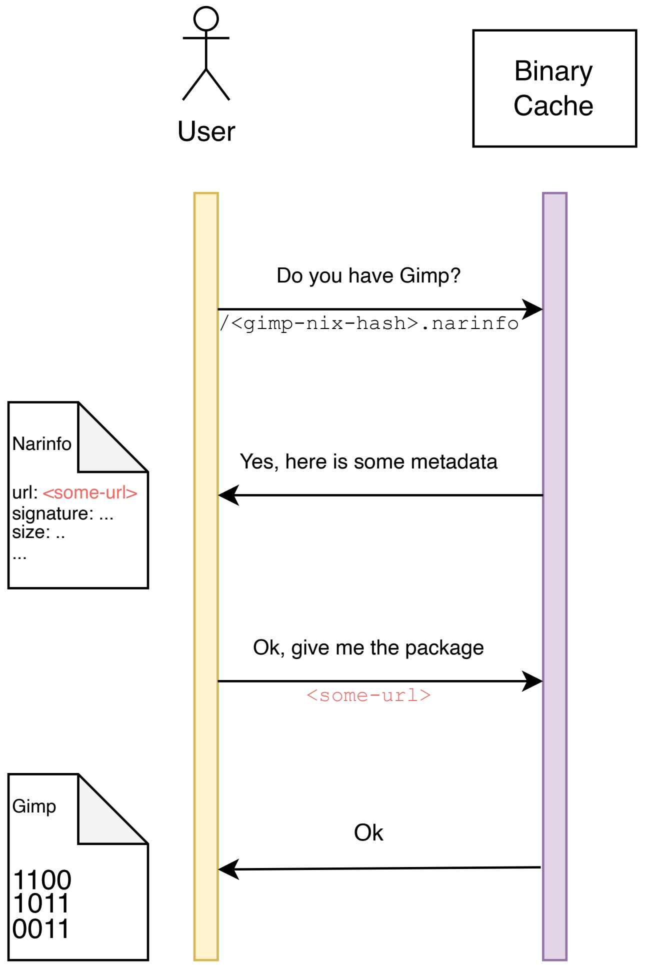
Binary Cache Protocol

- Occurs when executing:
  `nix build <some-package>`
- Protocol over HTTPS



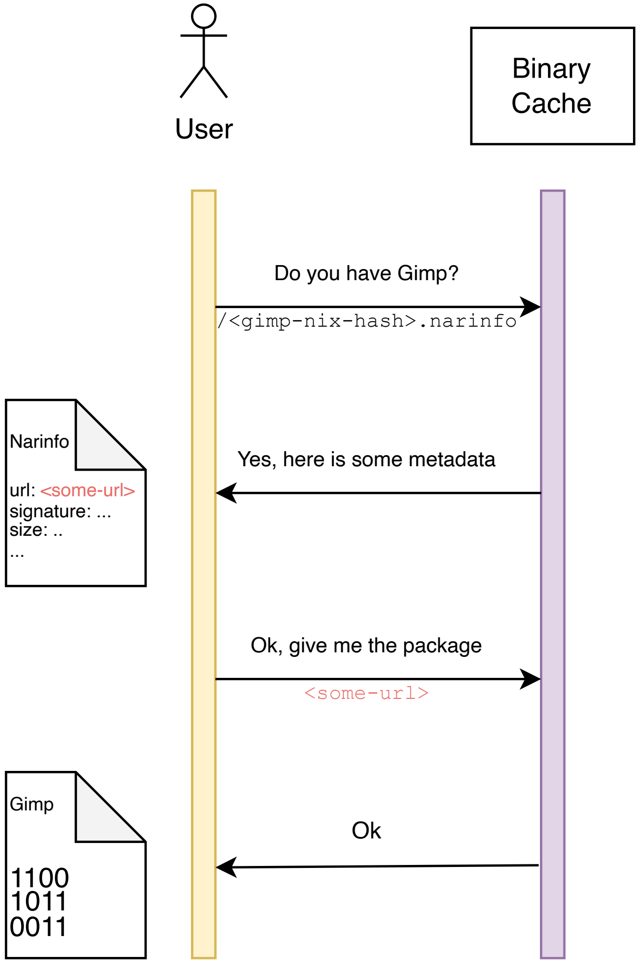
Binary Cache Protocol

- Occurs when executing:
  `nix build <some-package>`
- Protocol over HTTPS
- User asks for **Narinfo**, which contains URL to package contents



Binary Cache Protocol

- Occurs when executing: `nix build <some-package>`
- Protocol over HTTPS
- User asks for **Narinfo**, which contains URL to package contents
- Receives package in the **Nix Archive** (NAR) format



Git

- Advertised as distributed version control system

Git

- Advertised as distributed version control system
- Instead a tool for:
  - Manipulation of a **directed acyclic graph** (DAG)

Git

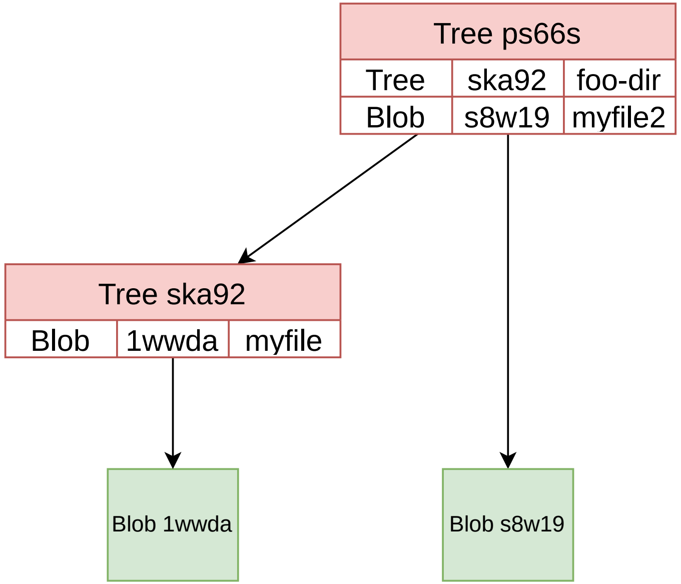
- Advertised as distributed version control system
- Instead a tool for:
  - Manipulation of a **directed acyclic graph** (DAG)
  - **Content-addressable objects**

Git

- Advertised as distributed version control system
- Instead a tool for:
  - Manipulation of a **directed acyclic graph** (DAG)
  - **Content-addressable objects**
  - **Replication** of these objects across repositories

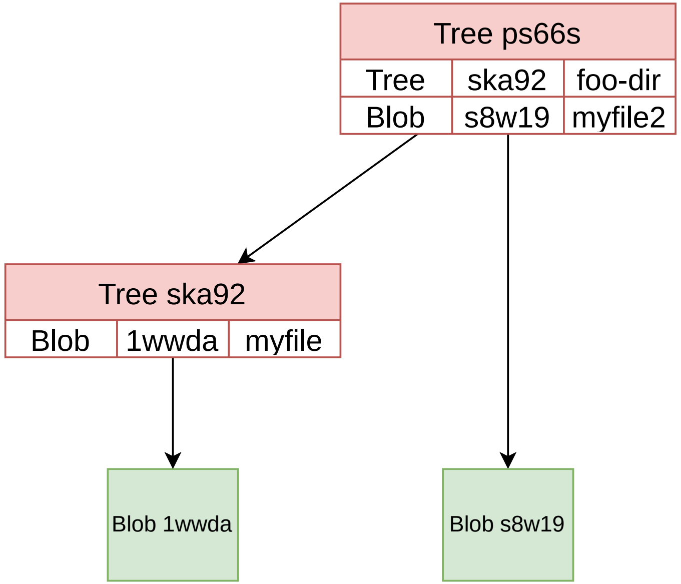
Git Objects: Blob and Tree

- **Blob**: Sequence of bytes, usually stores files



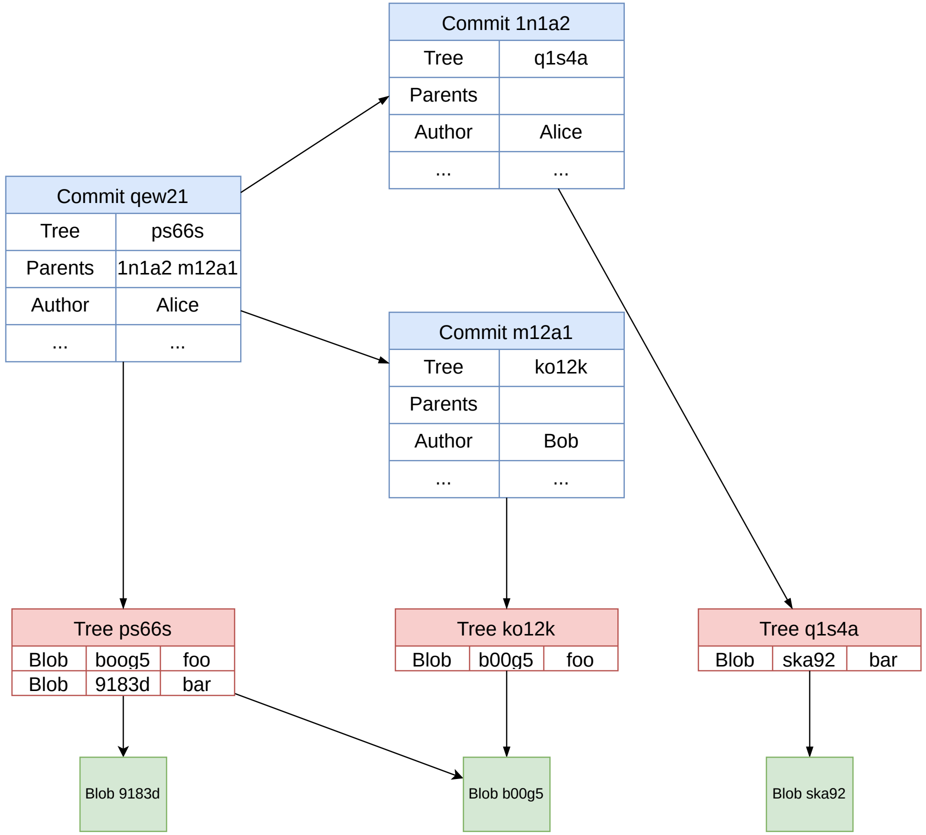
Git Objects: Blob and Tree

- **Blob**: Sequence of bytes, usually stores files
- **Tree**: Collection of pointers to blobs or trees.



Git Objects: Commit

- **Commit** contains:
  - Pointer to **exactly one tree**



Commit qew21	
Tree	ps66s
Parents	1n1a2 m12a1
Author	Alice
...	...

Commit 1n1a2	
Tree	q1s4a
Parents	
Author	Alice
...	...

Commit m12a1	
Tree	ko12k
Parents	
Author	Bob
...	...

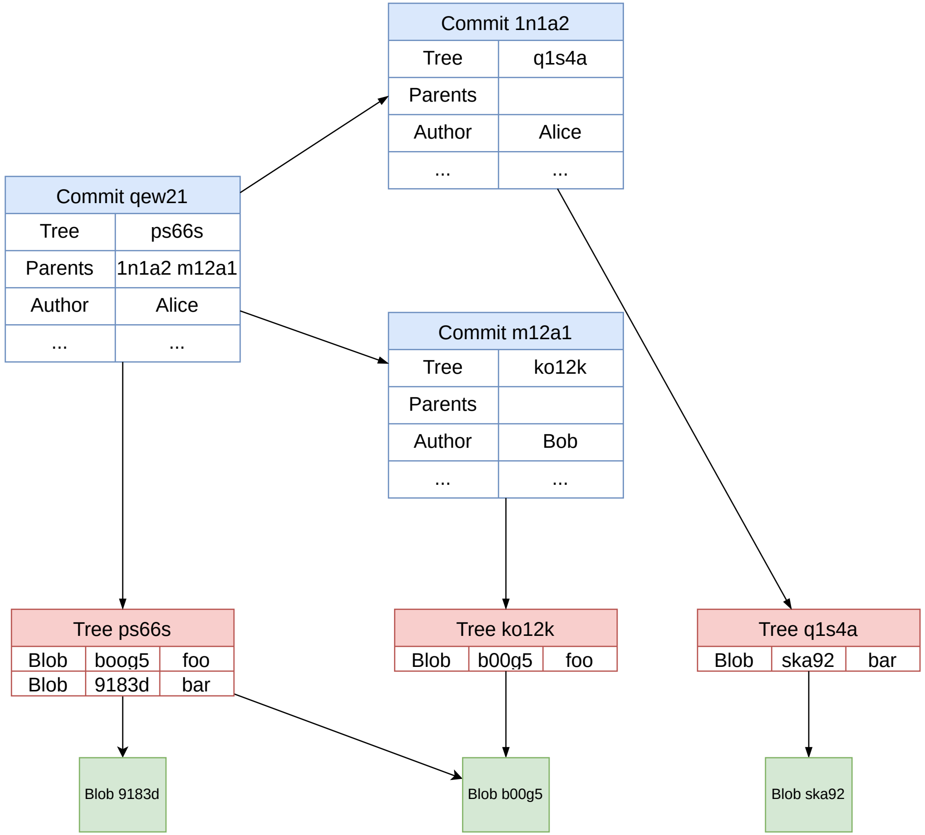
Tree ps66s		
Blob	boog5	foo
Blob	9183d	bar

Tree ko12k		
Blob	b00g5	foo

Tree q1s4a		
Blob	ska92	bar

Git Objects: Commit

- **Commit** contains:
  - Pointer to **exactly one tree**
  - **Parent Pointers**: Pointers to other commits



Commit qew21	
Tree	ps66s
Parents	1n1a2 m12a1
Author	Alice
...	...

Commit 1n1a2	
Tree	q1s4a
Parents	
Author	Alice
...	...

Commit m12a1	
Tree	ko12k
Parents	
Author	Bob
...	...

Tree ps66s		
Blob	boog5	foo
Blob	9183d	bar

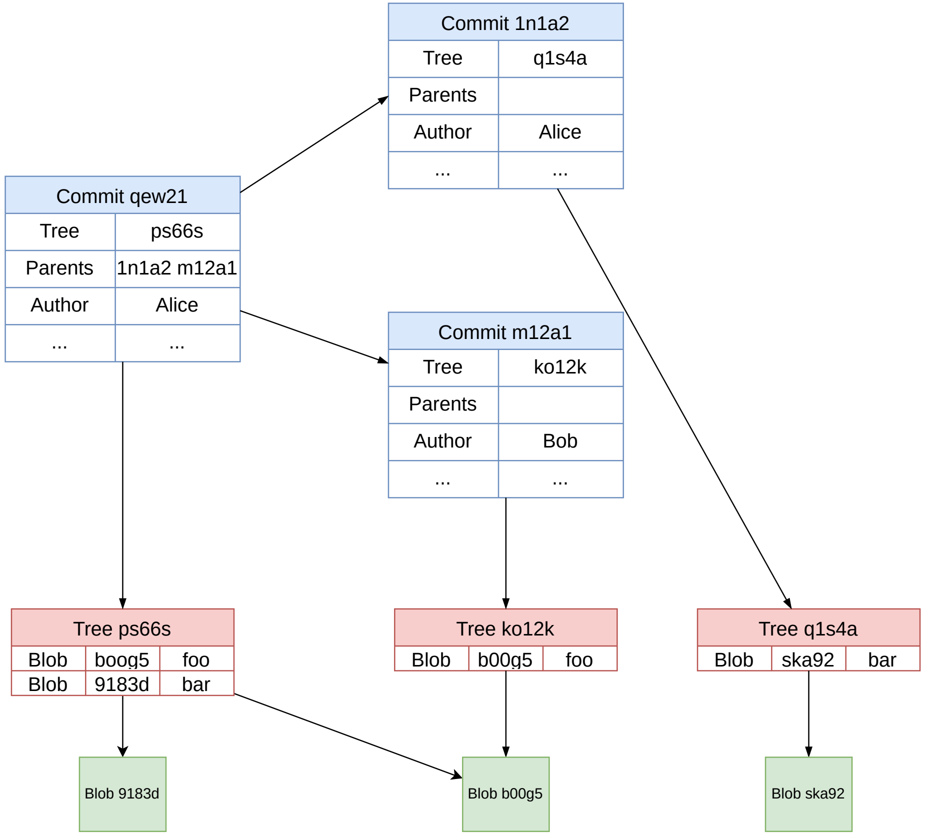
Tree ko12k		
Blob	b00g5	foo

Tree q1s4a		
Blob	ska92	bar

2.2. Git

Git Objects: Commit

- **Commit** contains:
  - Pointer to **exactly one tree**
  - **Parent Pointers**: Pointers to other commits
  - Author (name and mail)
  - Timestamp
  - Message



Commit 1n1a2	
Tree	q1s4a
Parents	
Author	Alice
...	...

Commit qew21	
Tree	ps66s
Parents	1n1a2 m12a1
Author	Alice
...	...

Commit m12a1	
Tree	ko12k
Parents	
Author	Bob
...	...

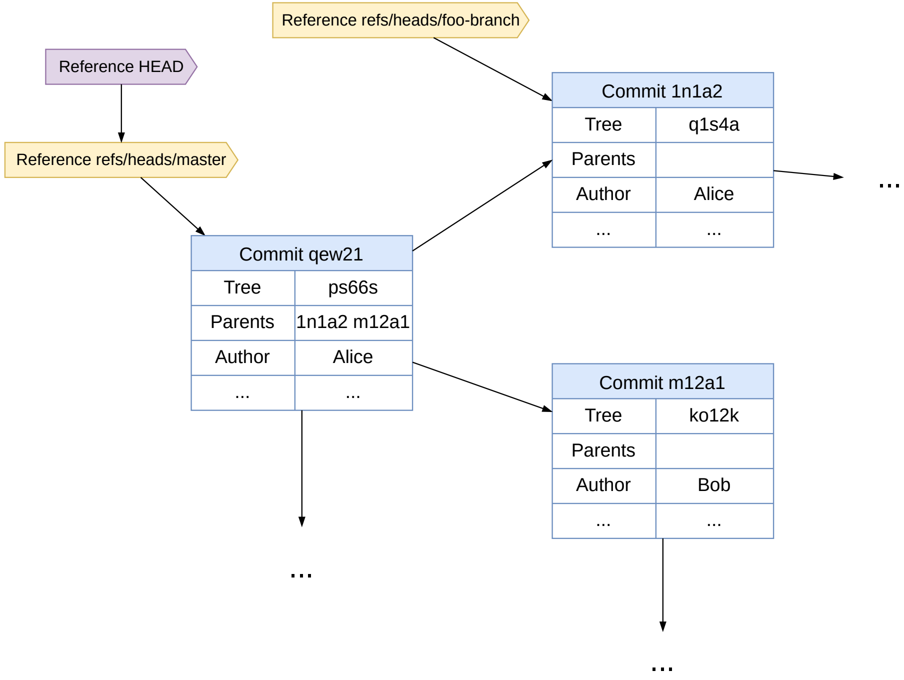
Tree ps66s		
Blob	boog5	foo
Blob	9183d	bar

Tree ko12k		
Blob	b00g5	foo

Tree q1s4a		
Blob	ska92	bar

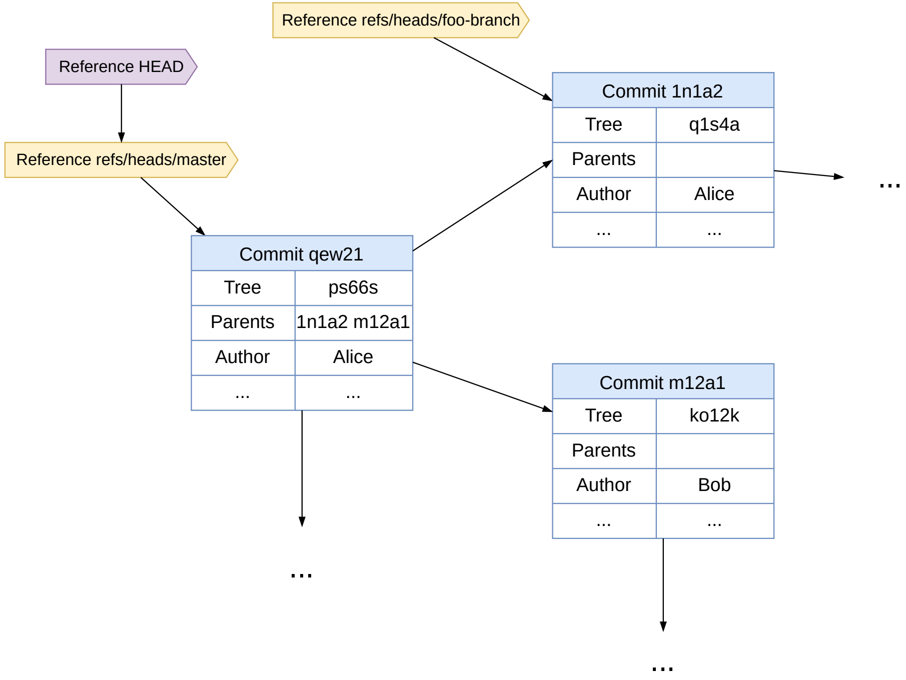
Git Objects: Reference

- **References**: Pointer to Git objects



Git Objects: Reference

- **References**: Pointer to Git objects
- **Direct Reference**: Points to blobs, trees, commits (e.g. branches and tags)



Reference refs/heads/foo-branch

Reference HEAD

Reference refs/heads/master

Commit qew21	
Tree	ps66s
Parents	1n1a2 m12a1
Author	Alice
...	...

Commit 1n1a2	
Tree	q1s4a
Parents	
Author	Alice
...	...

Commit m12a1	
Tree	ko12k
Parents	
Author	Bob
...	...

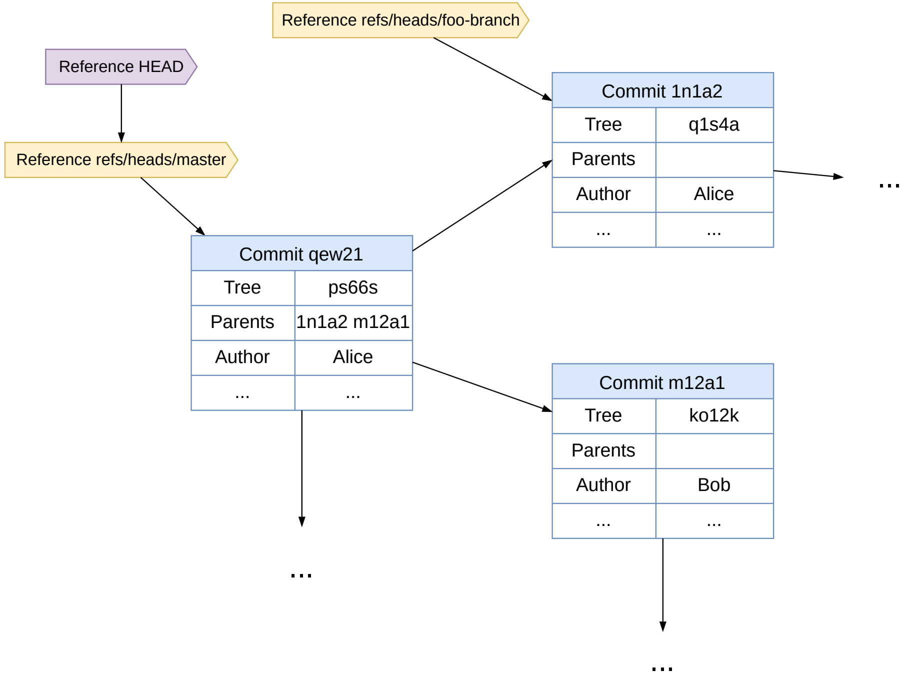
...

...

...

Git Objects: Reference

- **References**: Pointer to Git objects
- **Direct Reference**: Points to blobs, trees, commits (e.g. branches and tags)
- **Symbolic Reference**: Point to direct references (e.g. HEAD)



Git Objects

- Blobs, trees and commits are **immutable**

Git Objects

- Blobs, trees and commits are **immutable**
- Blobs, trees and commits are **content-addressed** (stored in `.git/objects`)
- References are mutable and identified by a given name (stored in `.git/refs`)

Replication

- Synchronize objects with `git fetch <refspec>`
- Specify objects with **refspecs**:
  - Constructed as `<remote_references>:<local_references>`
  - Copies the specified remote references
  - **Downloads all objects reachable** from the specified references
  - E.g. the command `git fetch refs/foo:refs/foo` copies refs/foo and downloads all object reachable from it

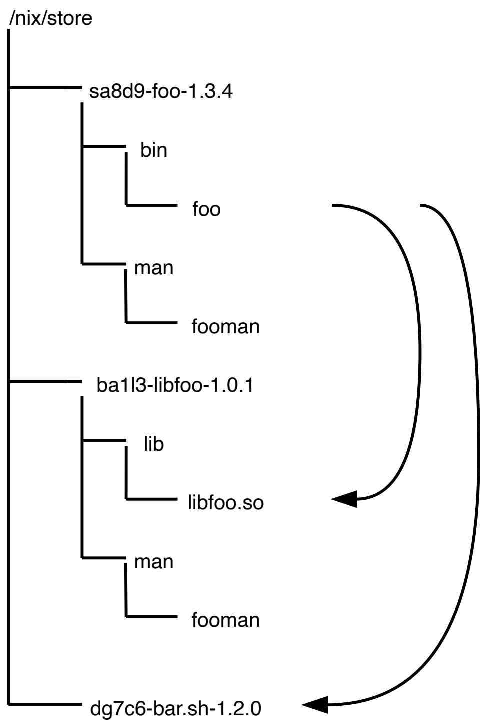
3. Design

3.1. Mapping Nix to Git

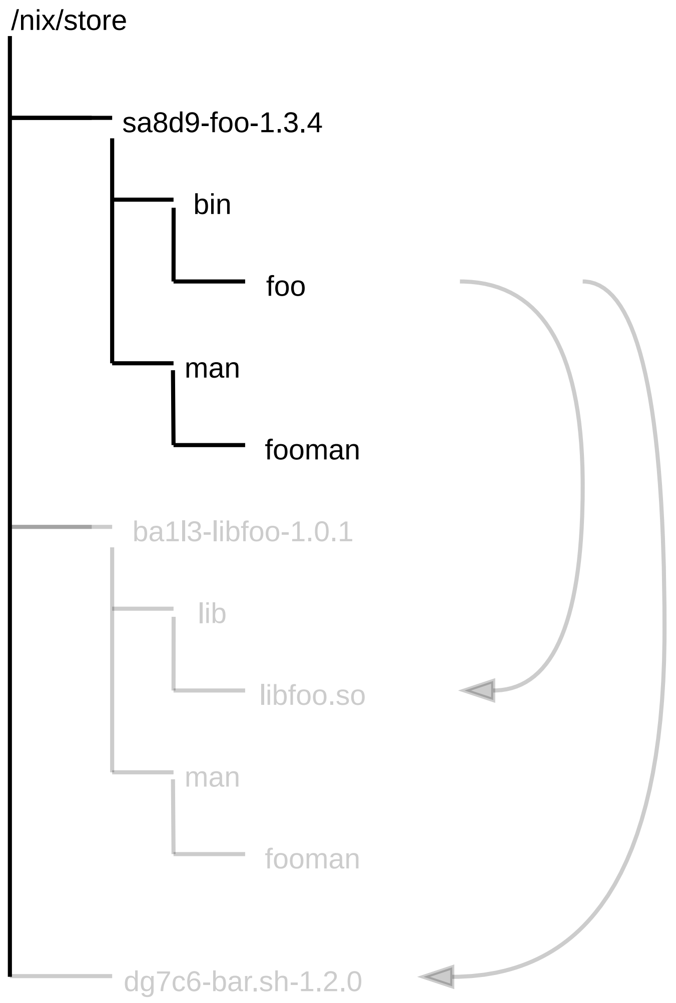
Nix to Git

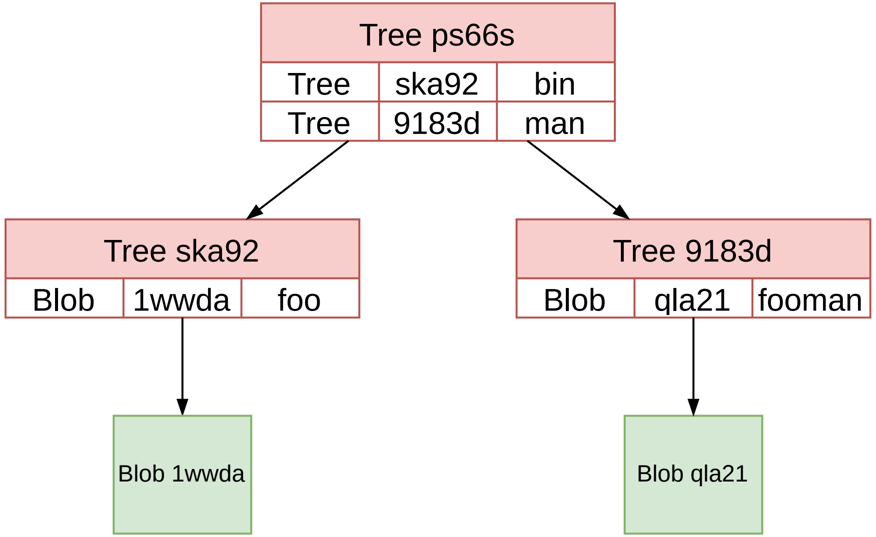
- Goal: Store Nix packages in a Git database.
- Map files to blobs
- Map directories to trees
- Model dependencies between packages using commit pointers

3.1. Mapping Nix to Git

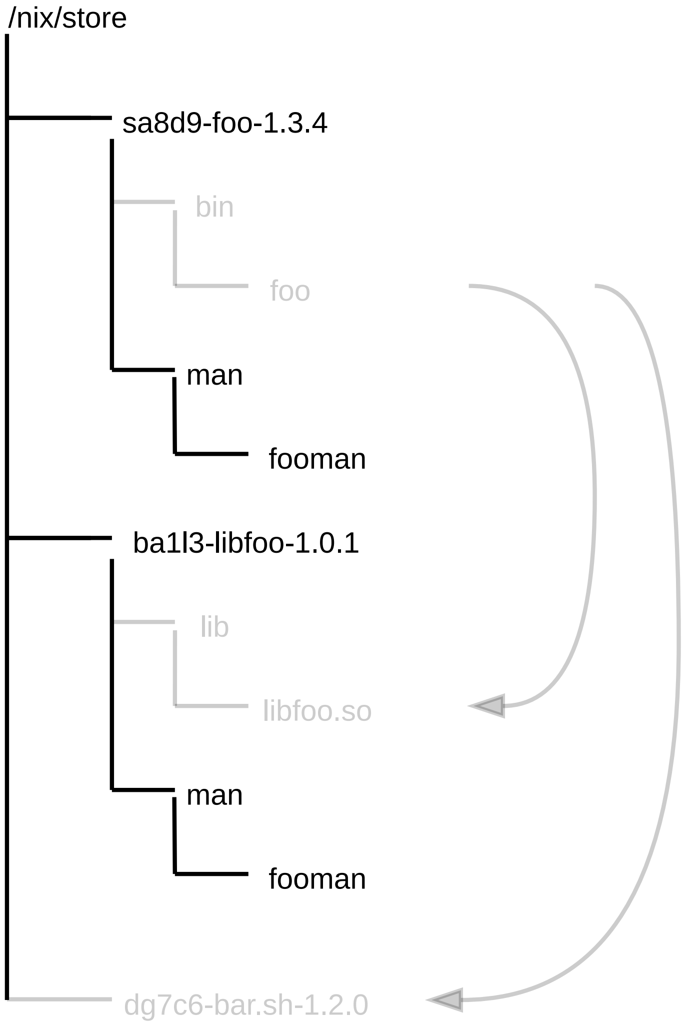


3.1. Mapping Nix to Git

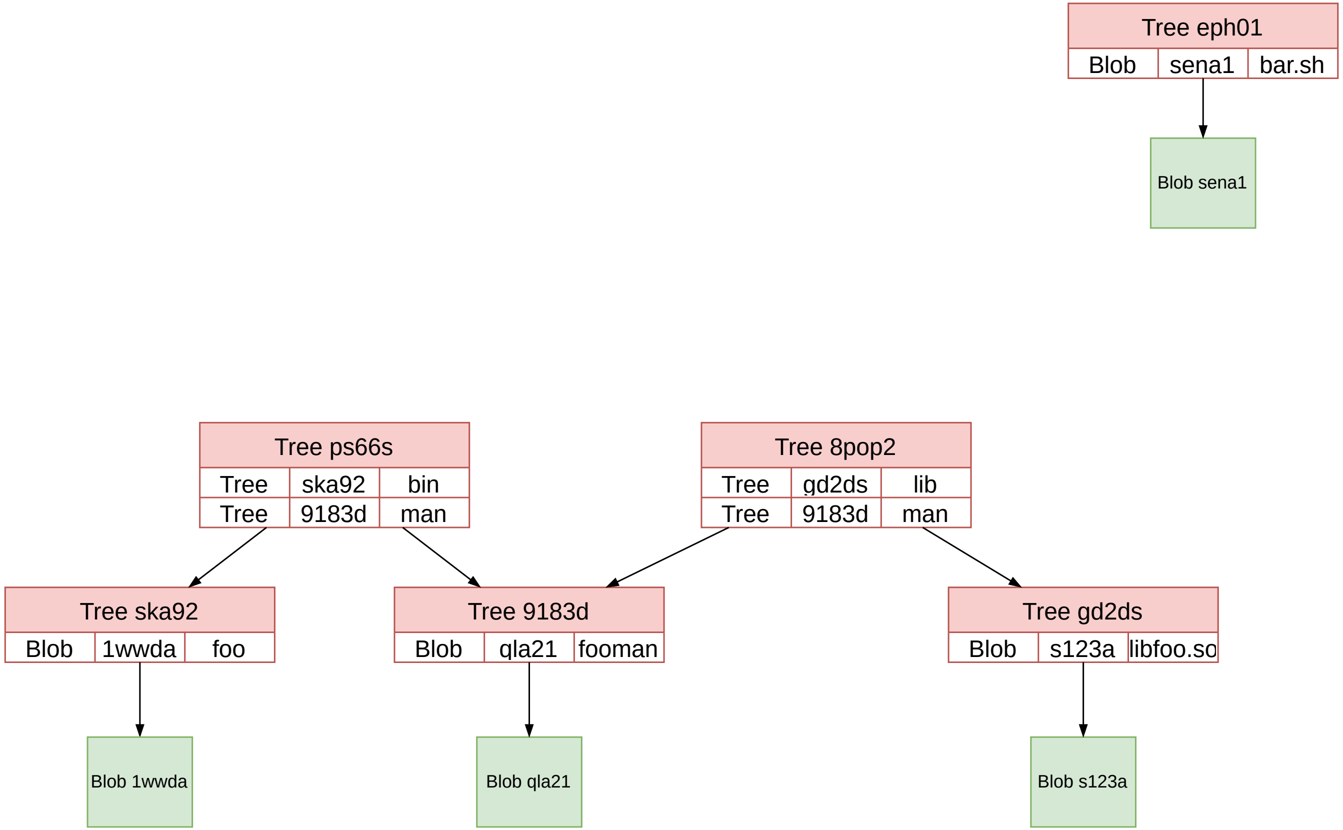




3.1. Mapping Nix to Git



3.1. Mapping Nix to Git



Tree eph01

Blob	sena1	bar.sh

Blob sena1

Tree ps66s

Tree	ska92	bin
Tree	9183d	man

Tree 8pop2

Tree	gd2ds	lib
Tree	9183d	man

Tree ska92

Blob	1wwda	foo

Tree 9183d

Blob	qla21	fooman

Tree gd2ds

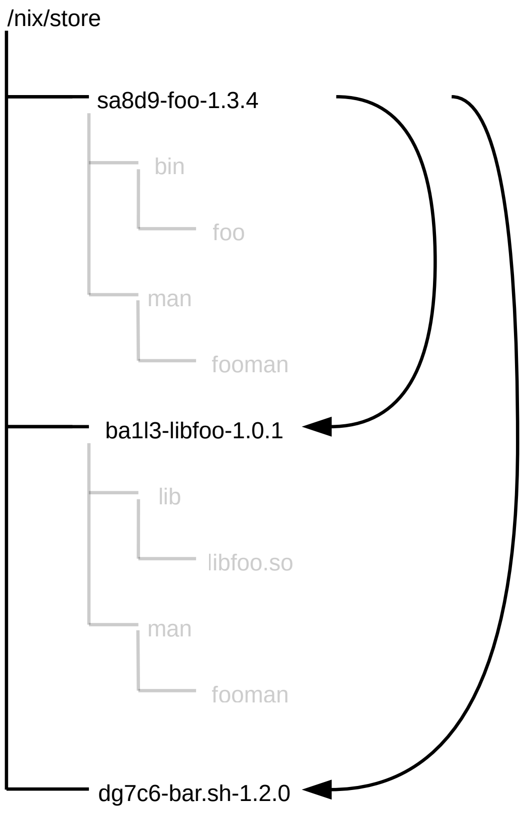
Blob	s123a	libfoo.so

Blob 1wwda

Blob qla21

Blob s123a

3.1. Mapping Nix to Git



/nix/store

sa8d9-foo-1.3.4

bin

foo

man

fooman

ba1l3-libfoo-1.0.1

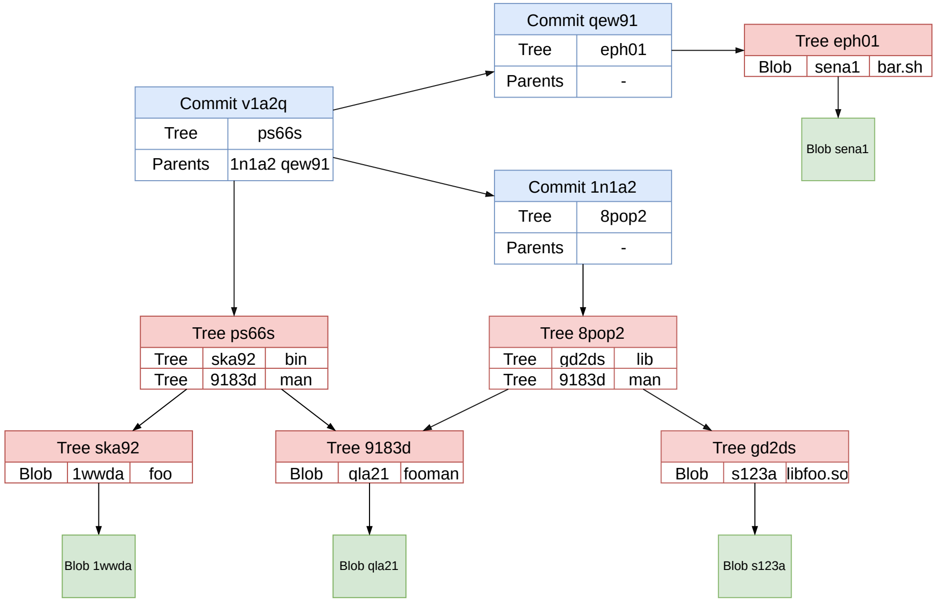
lib

libfoo.so

man

fooman

dg7c6-bar.sh-1.2.0



Commit qew91

Tree	eph01
Parents	-

Tree eph01

Blob	sena1	bar.sh

Blob sena1

Commit v1a2q

Tree	ps66s
Parents	1n1a2 qew91

Commit 1n1a2

Tree	8pop2
Parents	-

Tree ps66s

Tree	ska92	bin
Tree	9183d	man

Tree 8pop2

Tree	gd2ds	lib
Tree	9183d	man

Tree ska92

Blob	1wwda	foo

Tree 9183d

Blob	qla21	fooman

Tree gd2ds

Blob	s123a	libfoo.so

Blob 1wwda

Blob qla21

Blob s123a

Dependency Management

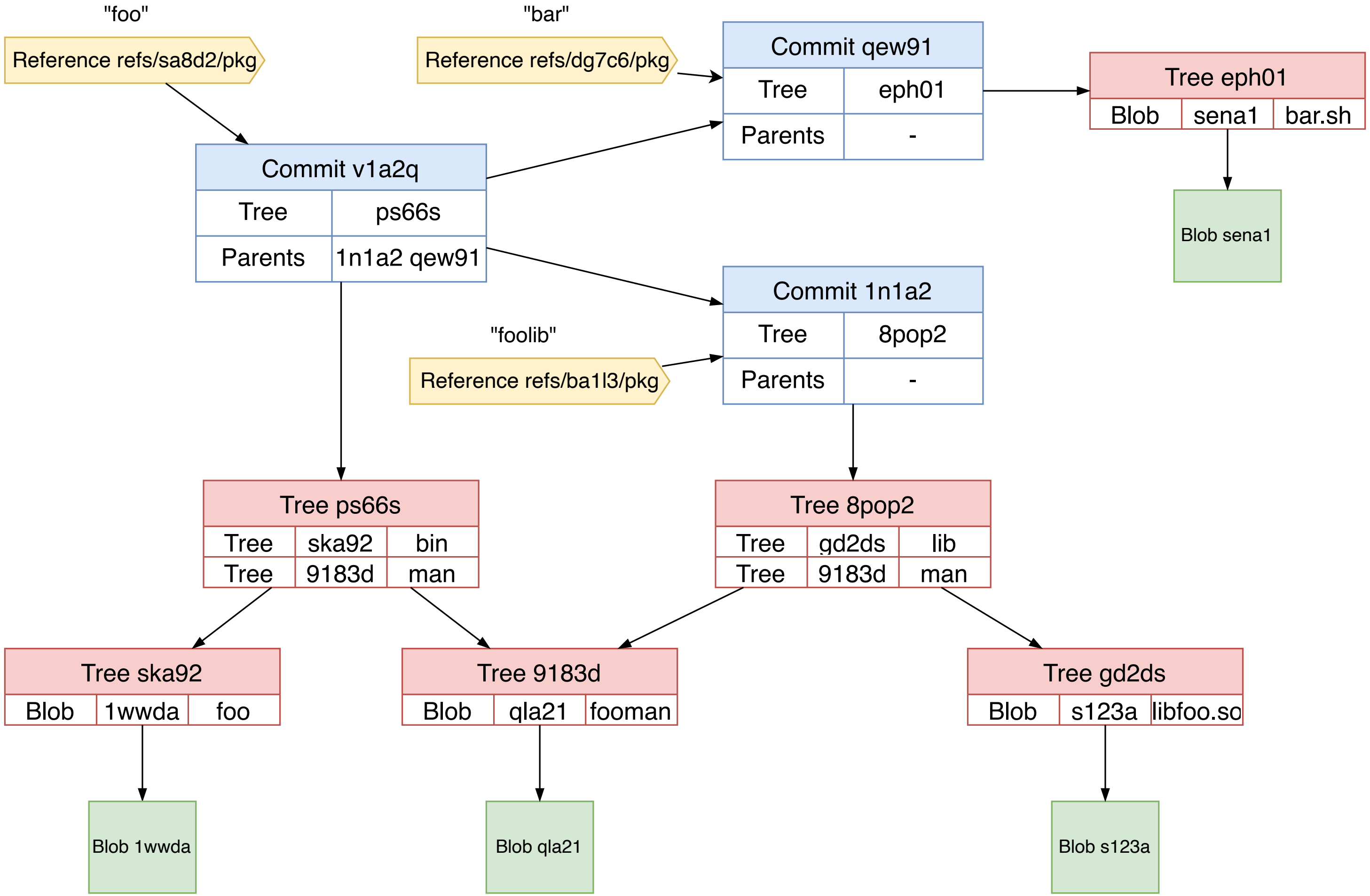
- Every package is associated with exactly one commit object
- Parents of the commit are dependencies of the package
- To ensure a global bijective mapping between commit hash and package:
  Set commit message, timestamp, and author field to constant values

3.2. Dependency Management

Dependency Management

- Every package is associated with exactly one commit object
- Parents of the commit are dependencies of the package
- To ensure a global bijective mapping between commit hash and package:
  Set commit message, timestamp, and author field to constant values
- Maintain mapping between Nix hashes and commit hashes using references in the form:
  `refs/<nix-hash>/pkg`

3.2. Dependency Management



"foo"

Reference refs/sa8d2/pkg

"bar"

Reference refs/dg7c6/pkg

"foolib"

Reference refs/ba1l3/pkg

Commit qew91	
Tree	eph01
Parents	-

Tree eph01		
Blob	sena1	bar.sh

Blob sena1

Commit v1a2q	
Tree	ps66s
Parents	1n1a2 qew91

Commit 1n1a2	
Tree	8pop2
Parents	-

Tree ps66s		
Tree	ska92	bin
Tree	9183d	man

Tree 8pop2		
Tree	gd2ds	lib
Tree	9183d	man

Tree ska92		
Blob	1wwda	foo

Tree 9183d		
Blob	qla21	fooman

Tree gd2ds		
Blob	s123a	libfoo.so

Blob 1wwda

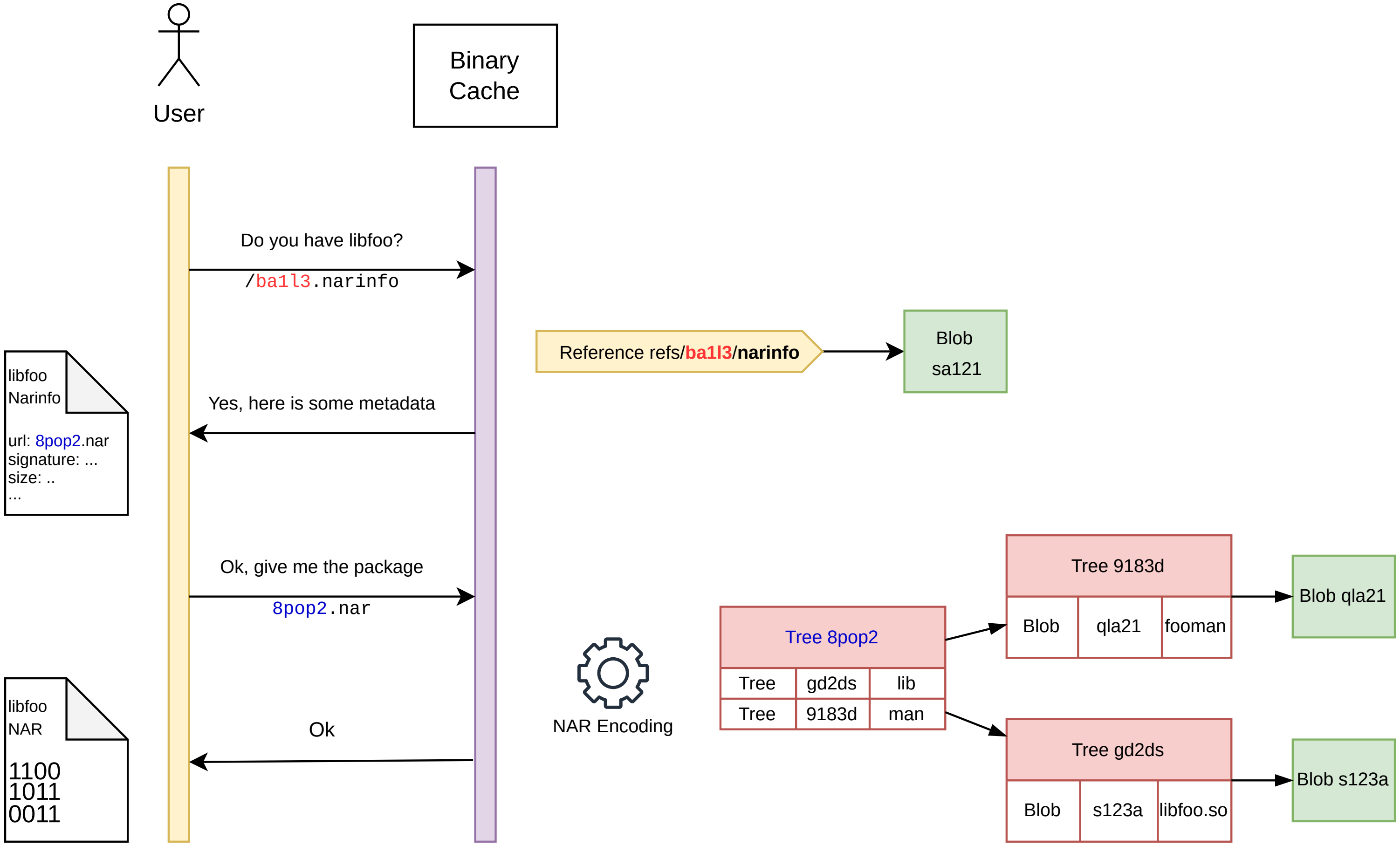
Blob qla21

Blob s123a

Binary Cache Protocol

- Everytime a package is added to the Git database, the Narinfo is constructed and the reference `/refs/<nix-hash>/narinfo` points to it
- The server transforms package into Nix Archive (NAR) and streams it to the user

3.3. Binary Cache Protocol



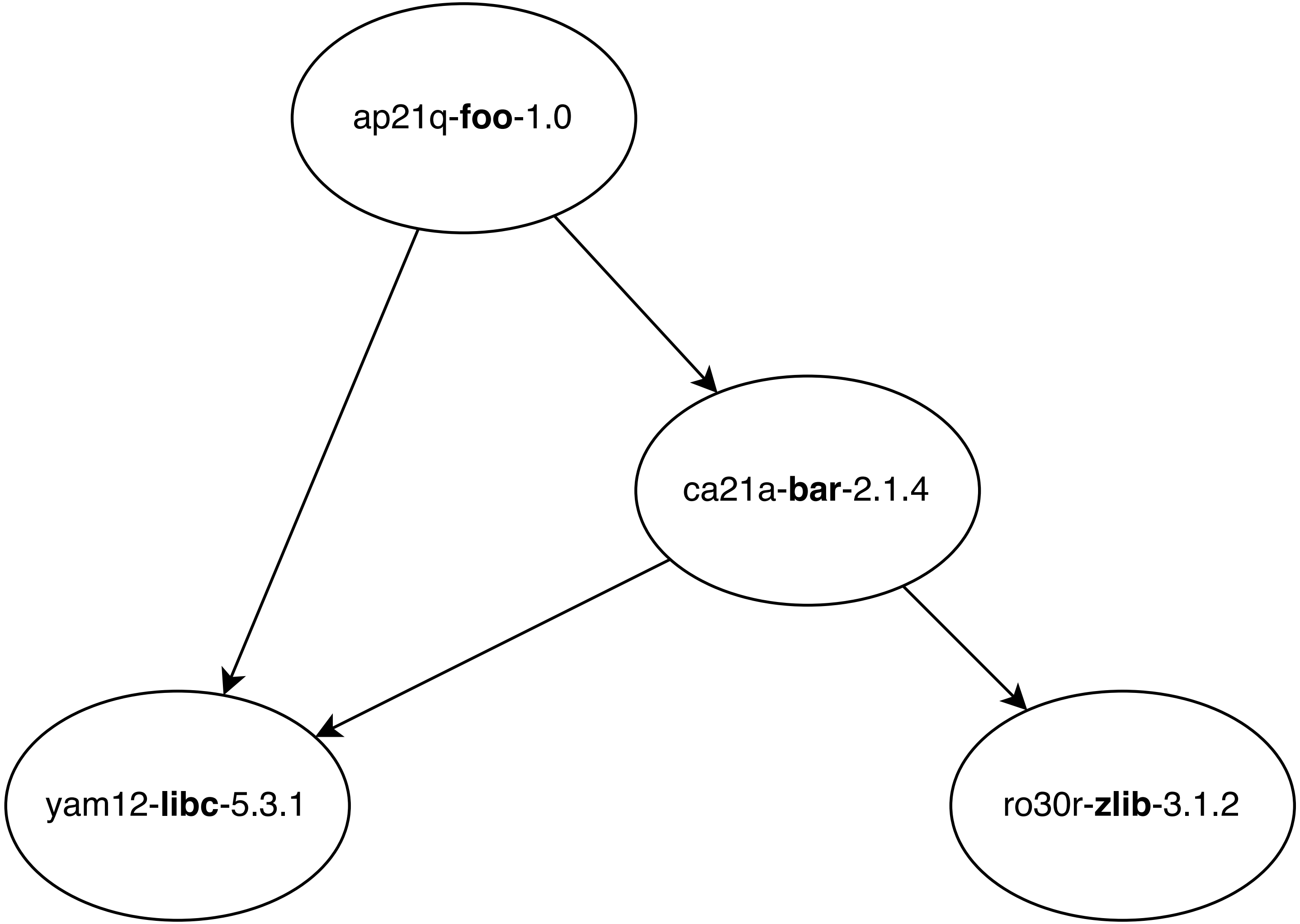
Replication

- There is no policy (yet) on when a package is added to the cache
- Packages are either added by **fetching commits from peers** or **constructed using Nix interface**

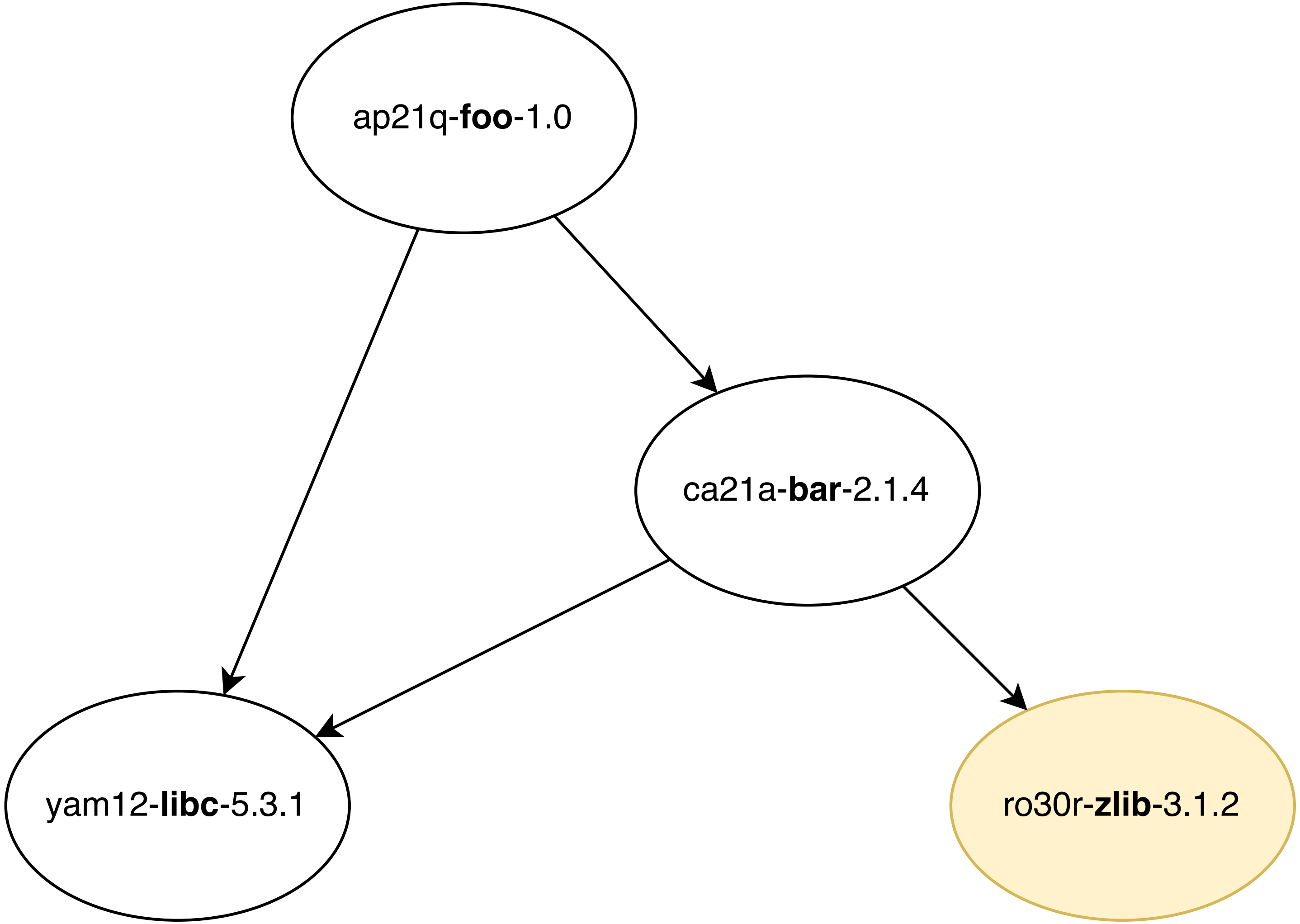
Replication with Nix interface

- Explore the dependency graph of a package in a **depth first search** manner:
  - Iterate through dependencies
  - Fully *process* one dependency, including all its dependencies, before moving to the next dependency in the list
  - *Processing* includes:
    - Fetch NAR using Nix interface and decode it to Git objects
    - Build the Narinfo
    - Create the package commit and set references

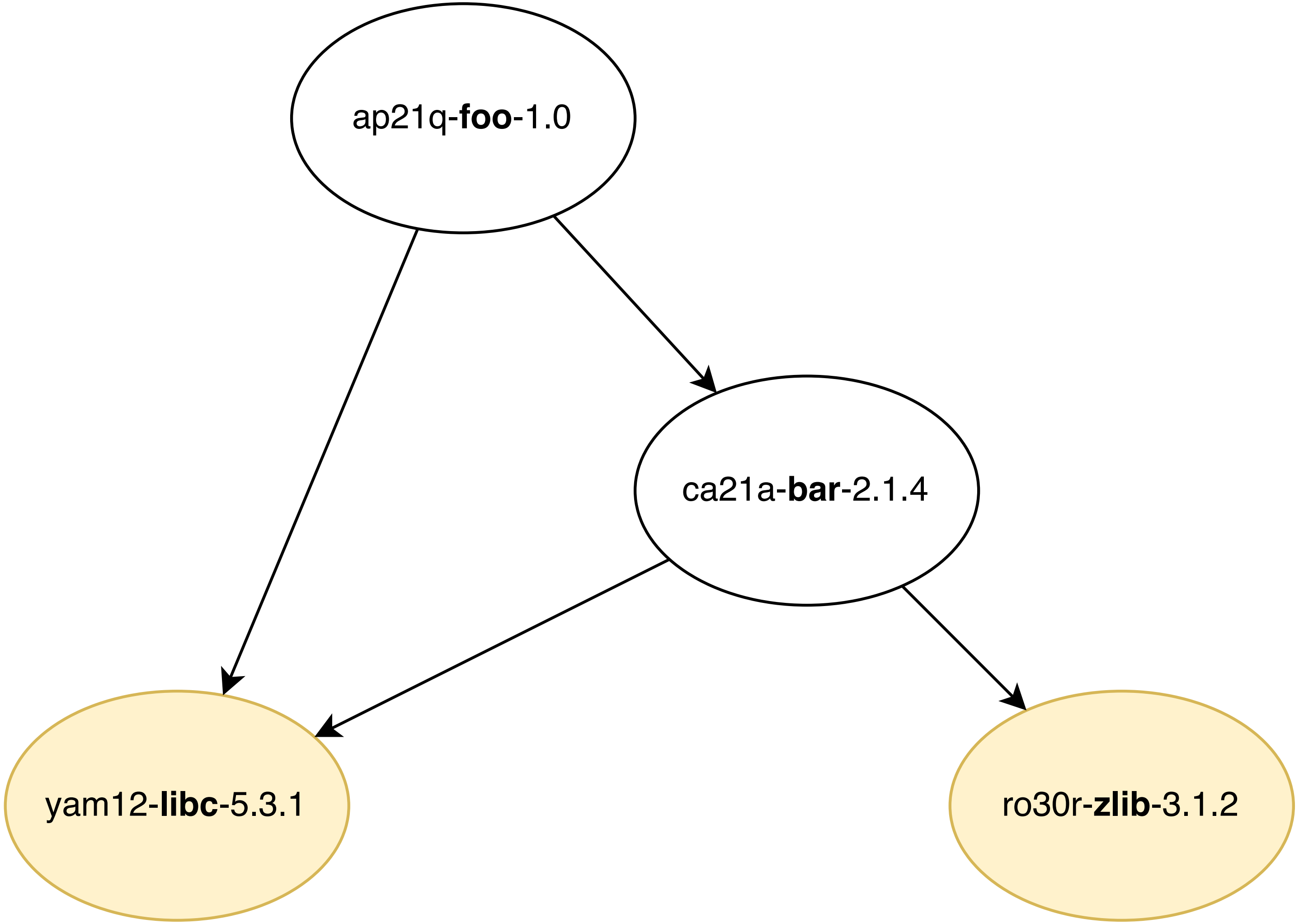
3.4. Replication

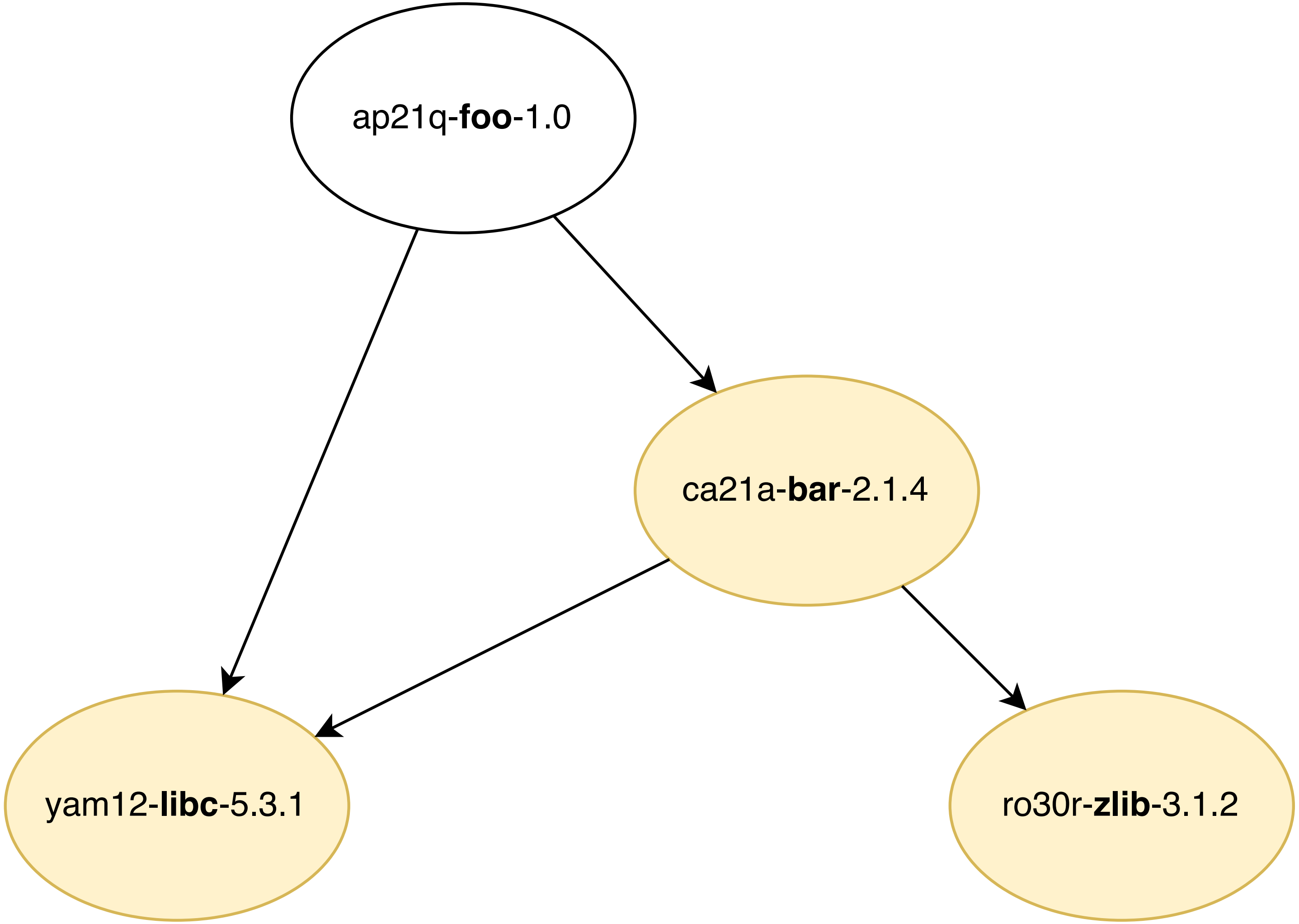


3.4. Replication

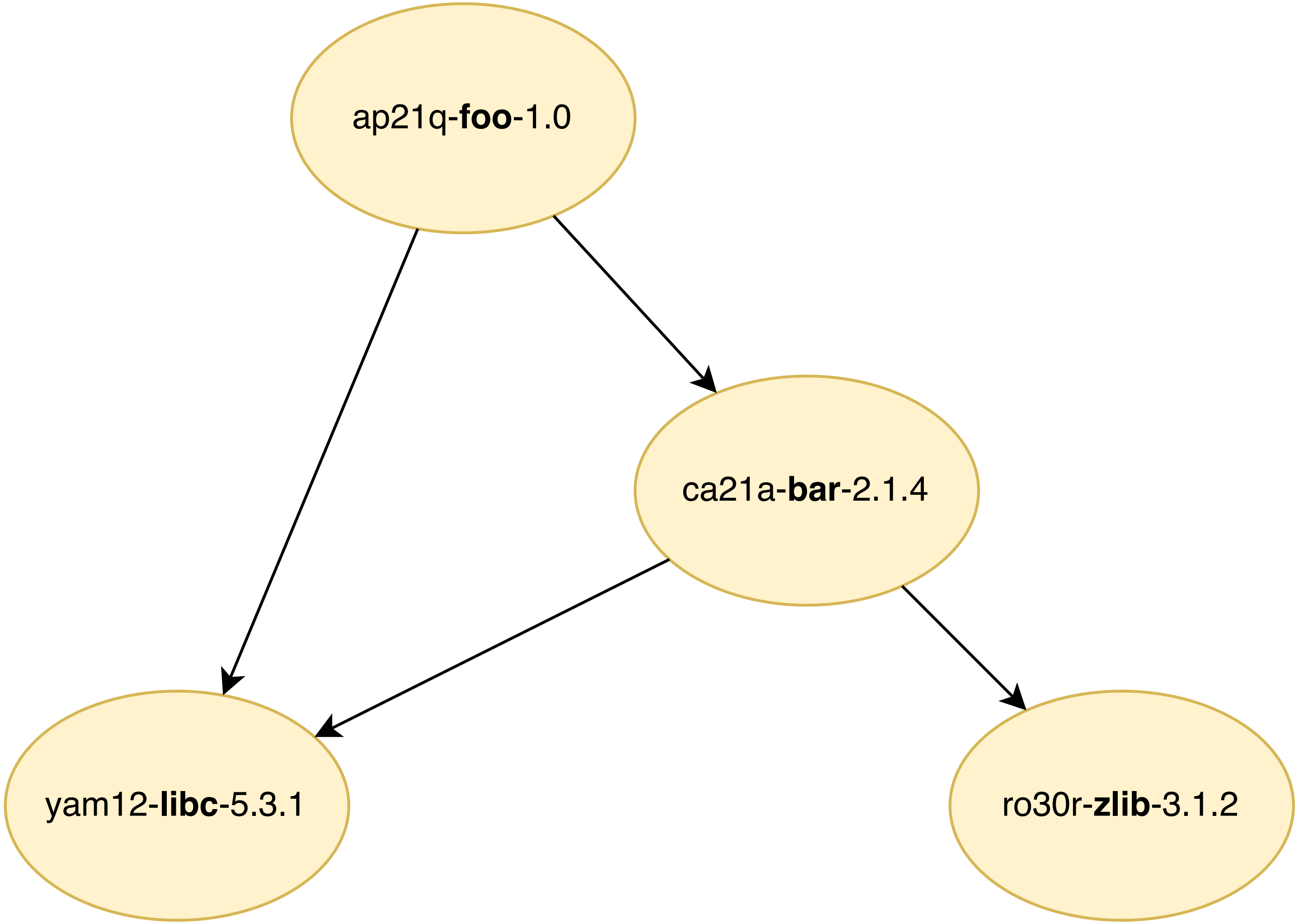


3.4. Replication




ap21q-**foo**-1.0
ca21a-**bar**-2.1.4
yam12-**libc**-5.3.1
ro30r-**zlib**-3.1.2

3.4. Replication



Replication with Gachix peers

- Do `git fetch refs/<h>/*:refs/<h>/*` where `<h>` is the hash of the requested package
  → fetches the reference `refs/<h>` and all objects reachable from it

Replication with Gachix peers

- Do `git fetch refs/<h>/*:refs/<h>/*` where `<h>` is the hash of the requested package
  → fetches the reference `refs/<h>` and all objects reachable from it
- For every dependency fetch the missing reference object

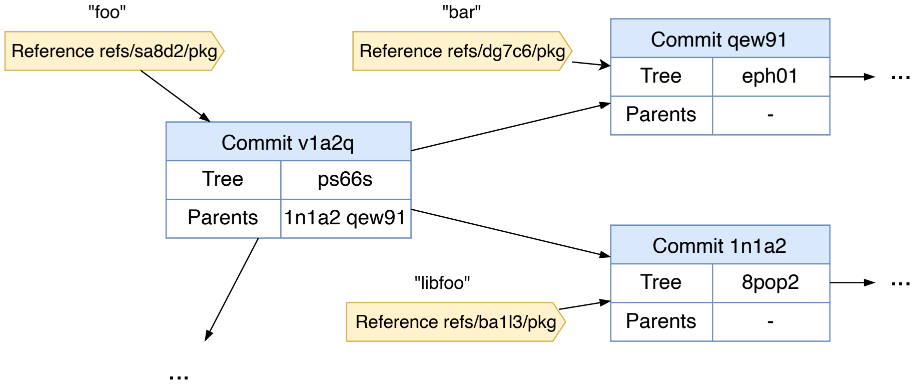
Replication with Gachix peers

- Do `git fetch refs/<h>/*:refs/<h>/*` where `<h>` is the hash of the requested package
  → fetches the reference `refs/<h>` and all objects reachable from it
- For every dependency fetch the missing reference object
- This can be done in a breadth first search manner

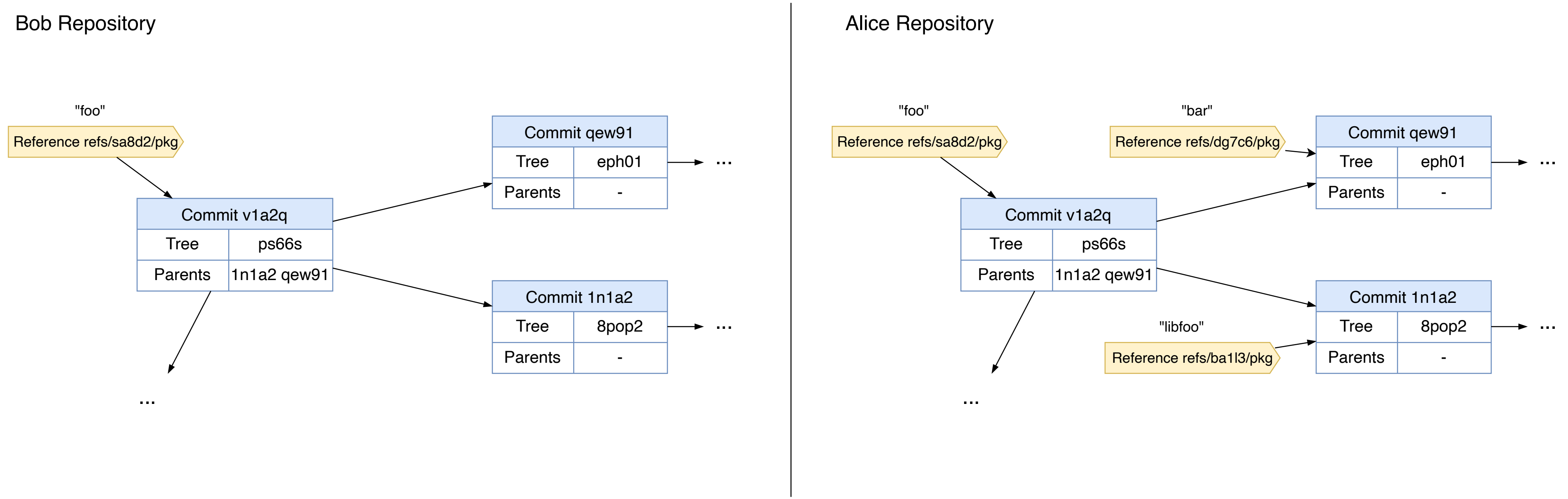
3.4. Replication

Bob Repository

Alice Repository

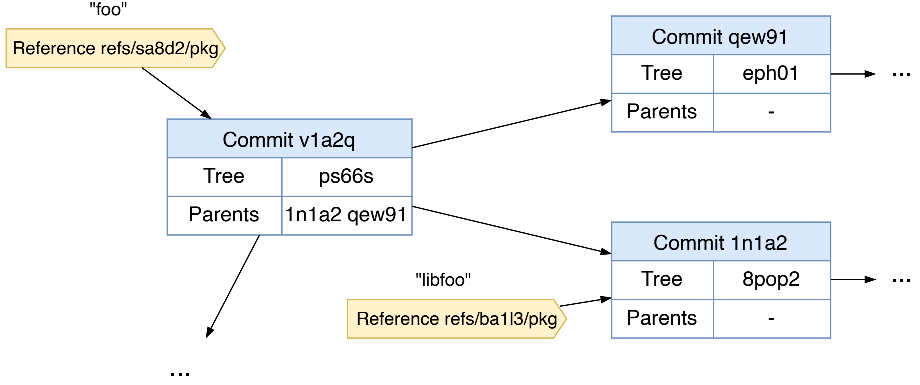


3.4. Replication



3.4. Replication

Bob Repository



"foo"

Reference refs/sa8d2/pkg

Commit v1a2q	
Tree	ps66s
Parents	1n1a2 qew91

Commit qew91	
Tree	eph01
Parents	-

Commit 1n1a2	
Tree	8pop2
Parents	-

"libfoo"

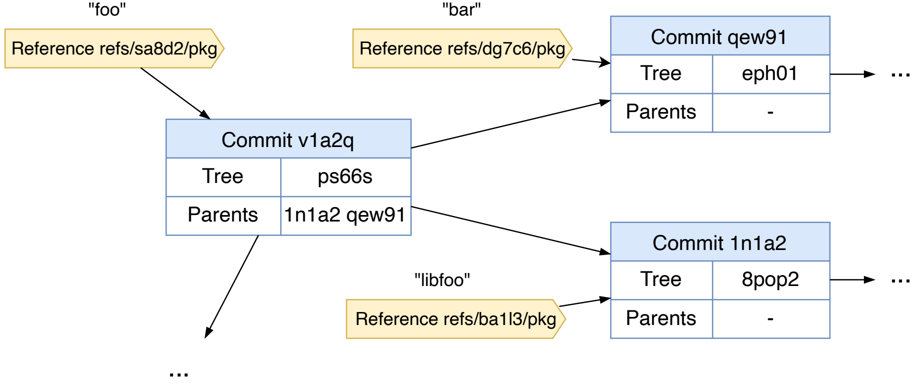
Reference refs/ba1l3/pkg

...

...

...

Alice Repository



"foo"

Reference refs/sa8d2/pkg

"bar"

Reference refs/dg7c6/pkg

Commit v1a2q	
Tree	ps66s
Parents	1n1a2 qew91

Commit qew91	
Tree	eph01
Parents	-

Commit 1n1a2	
Tree	8pop2
Parents	-

"libfoo"

Reference refs/ba1l3/pkg

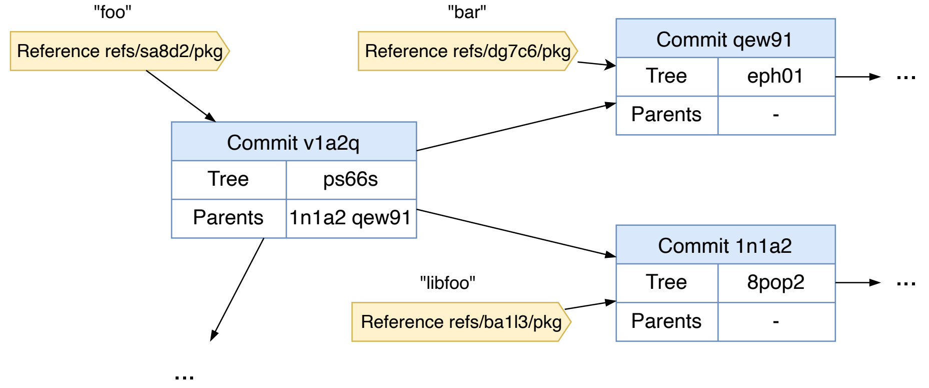
...

...

...

3.4. Replication

Bob Repository



"foo"

Reference refs/sa8d2/pkg

"bar"

Reference refs/dg7c6/pkg

"libfoo"

Reference refs/ba1l3/pkg

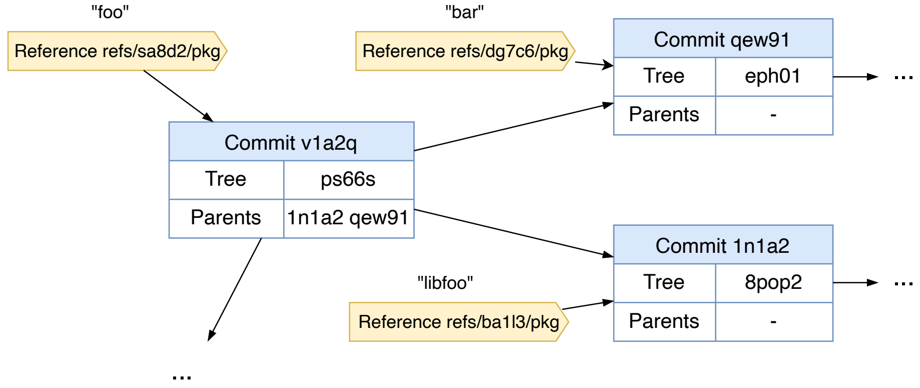
Commit v1a2q	
Tree	ps66s
Parents	1n1a2 qew91

Commit qew91	
Tree	eph01
Parents	-

Commit 1n1a2	
Tree	8pop2
Parents	-

...

Alice Repository



"foo"

Reference refs/sa8d2/pkg

"bar"

Reference refs/dg7c6/pkg

"libfoo"

Reference refs/ba1l3/pkg

Commit v1a2q	
Tree	ps66s
Parents	1n1a2 qew91

Commit qew91	
Tree	eph01
Parents	-

Commit 1n1a2	
Tree	8pop2
Parents	-

...

4. Results

4.1. Results Overview

Results

1. Gachix achieves the **lowest median latency** but shows **slower average** performance.
2. Gachix is **more storage efficient** than other cache services.
3. Gachix can be **deployed on any Unix machine**, including on systems without Nix installed.
4. Gachix is **transparent**: It can be used with the Nix interface.

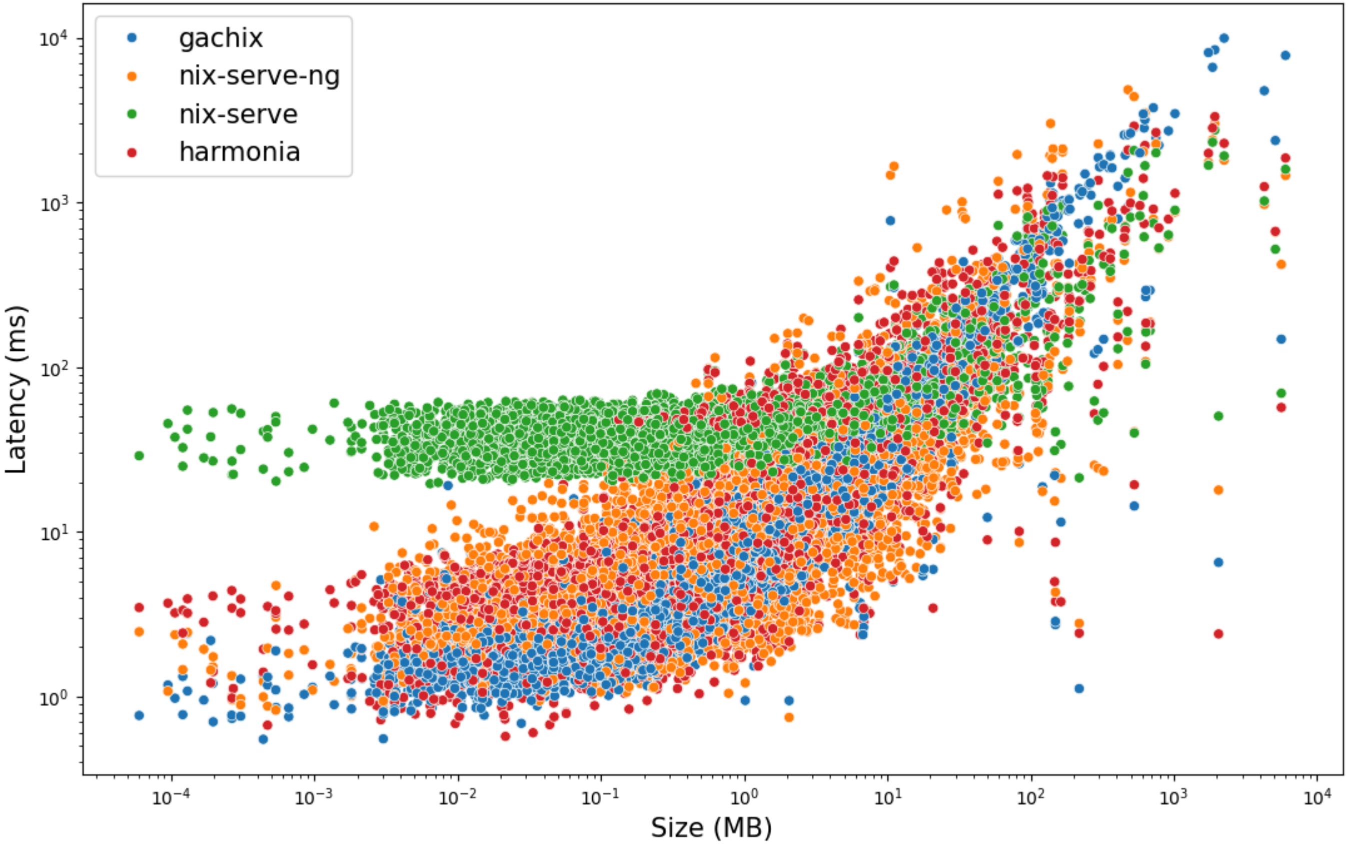
Methodology

1. Randomly select 650 packages from the Nixpkgs registry (from two releases)
2. Install them along with every required dependency → **5123 packages** in total
3. Add them to the Gachix cache
4. Start each cache service and **measure end-to-end latency** of NAR retrieval of each package

Results

Cache Service	Median	p95	p99	Max	Mean	Std
gachix	4.812	142.129	840.199	9931.217	49.347	327.198
harmonia	8.530	119.514	604.240	3316.529	41.912	146.475
nix-serve	42.063	101.205	447.337	2749.474	57.757	107.114
nix-serve-ng	7.689	105.879	616.550	4832.431	37.989	182.101

4.2. Package Retrieval Latency



Methodology

- Same as in the previous benchmark
- Add 5123 packages to the Nix store and to Gachix
- Measure size of `.git` and sum of the size of all packages in Nix store

4.3. Package Storage

Result

- Sum of package sizes in the Nix store is 77.87 GB
- Size of `.git` repository is 13.45 GB
- **Size reduction of 82.72%**
- 21.84% of Git objects had an indegree > 1

Thank you for your attention!