REUPNix: Reconfigurable and Updateable Embedded Systems
Sending stuff 🛰 into space 🚀 is expensive 💰
Sending stuff 🛰  into space 🚀  is expensive 💰

update
Sending stuff 🛰 into space 🚀 is expensive 💰

update

reconfigure
Sending stuff 🛰 into space 🚀 is expensive 💰

update

reconfigure

1 2 3
Sending stuff 🛰 into space 🚀 is expensive 💰

update

reconfigure

1 2 3 A B C
Sending stuff 🛰 into space 🚀 is expensive 💰

update

1 2 3 A B C

reconfigure

NixOS

REUPNix: Reconfigurable and Updateable Embedded Systems
Sending stuff 🛰 into space 🚀 is expensive 💰

update

reconfigure

1 2 3 A B C

NixOS 🖳 🖥 :
embedded

---

REUPNIX: Reconfigurable and Updateable Embedded Systems
Sending stuff 🛰 into space 🚀 is expensive 💰

update

1 2 3 A B

reconfigure

A B C

why?

NixOS 🖳 🖥 : embedded

embedded
NixOS & reUpNix

Nix

- Large base system → Shrink (~1.1 GiB → 155 MiB)
- OCI containers inefficient → Remove Docker daemon & deduplicate files across images
- Large system updates → Differential update transfer
- Bootloader update fragile → Make update atomic & Integrate reconfiguration
NixOS & reUpNix

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images
Large system updates ⇒ Differential update transfer
Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images
Large system updates ⇒ Differential update transfer
Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix

Nix

NixOS

/etc

sshd

systemd

NixOS reUpNix contribution

Large base system
⇒ Shrink (~1.1 GiB → 155 MiB)

OCI containers inefficient
⇒ Remove Docker daemon & deduplicate files across images

Large system updates
⇒ Differential update transfer

Bootloader update fragile
⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix

- Nix /nix/store
- systemd
- NixOS
- /etc
- sshd
- systemd

---

reUpNix contribution

- Large base system ➔ Shrink (~1.1 GiB → 155 MiB)
- OCI containers inefficient ➔ Remove Docker daemon & deduplicate files across images
- Large system updates ➔ Differential update transfer
- Bootloader update fragile ➔ Make update atomic & Integrate reconfiguration

---

REUPNix: Reconfigurable and Updateable Embedded Systems
NixOS & reUpNix

Nix

/nix/store

/etc

NixOS

sshhd

systemd

NixOS & reUpNix contribution

Large base system

⇒

Shrink (~1.1 GiB → 155 MiB)

OCI containers inefficient

⇒

Remove Docker daemon & deduplicate files across images

Large system updates

⇒

Differential update transfer

Bootloader update fragile

⇒

Make update atomic & Integrate reconfiguration

REUPNIX: Reconfigurable and Updateable Embedded Systems
NixOS & reUpNix

- Nix files (.nix)
- /nix/store
- systemd
- /etc
- sshd
- NixOS
- NixOS
- NixOS

**reUpNix contribution**

- Large base system
  - Shrink (~1.1 GiB → 155 MiB)
- OCI containers inefficient
  - Remove Docker daemon & deduplicate files across images
- Large system updates
  - Differential update transfer
- Bootloader update fragile
  - Make update atomic & Integrate reconfiguration

**REUPNix:** Reconfigurable and Updateable Embedded Systems
NixOS & reUpNix

- Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
- OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images
- Large system updates ⇒ Differential update transfer
- Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix

Nix

/nix/store

/etc

NixOS

sshd

systemd

/etc
NixOS & reUpNix

Nix

/nix/store

/etc

NixOS

sshd

systemd

/reproducible

NixOS

/etc

NixOS

/reproducible

Nix

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)

OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images

Large system updates ⇒ Differential update transfer

Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix

- Nix
  - /nix/store
  - /etc
  - NixOS
  - systemd
  - sshd
  - /etc

- flexible
- reproducible

ReUpNix contribution:
- Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
- OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images
- Large system updates ⇒ Differential update transfer
- Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration

REUPNix: Reconfigurable and Updateable Embedded Systems
NixOS & reUpNix

Nix

/nix/store

/etc

NixOS

sshd

systemd

/etc

NixOS

reproducible

flexible

reconfiguration

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)

OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images

Large system updates ⇒ Differential update transfer

Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix

Nix

flexible

reproducible

NixOS X

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
OCI containers inefficient ⇒ Remove Docker daemon & deduplicate files across images
Large system updates ⇒ Differential update transfer
Bootloader update fragile ⇒ Make update atomic & Integrate reconfiguration
NixOS & reUpNix contribution

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)

NixOS ×

flexible

reproducible

reconfiguration

Nix

/nix/store

/etc

NixOS

sshd

systemd

/etc

NixOS
NixOS & reUpNix

NixOS ×
Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
OCI containers inefficient ⇒ Remove Docker daemon
& deduplicate files across images

reUpNix contribution

NixOS

flexible

reproducible

Nix

/nix/store

/etc

NixOS

sshd

systemd

/etc

NixOS

ng
REUPNix: Reconfigurable and Updateable Embedded Systems
NixOS & reUpNix

**Flexible**

Nix

NixOS

/etc

sshd

NixOS

/etc

.nix

reproducible

Nix /nix/store

systemd

Nix /etc

reUpNix contribution

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)

OCI containers inefficient ⇒ Remove Docker daemon

Large system updates ⇒ & deduplicate files across images

Differential update transfer
NixOS & reUpNix

Nix

Flexible

Reproducible

NixOS

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
OCI containers inefficient ⇒ Remove Docker daemon
& deduplicate files across images
Large system updates ⇒ Differential update transfer
Bootloader update fragile ⇒ Make update atomic
& Integrate reconfiguration

reUpNix contribution

NixOS

/etc

NixOS

systemd

/etc

nix/store

flexible

reproducible

Nix

Large base system
OCI containers inefficient
Large system updates
Bootloader update fragile

NixOS

shaddr
NixOS & reUpNix

Nix

flexible

reproducible

NixOS

Large base system ⇒ Shrink (~1.1 GiB → 155 MiB)
OCI containers inefficient ⇒ Remove Docker daemon
& deduplicate files across images
Large system updates ⇒ Differential update transfer
Bootloader update fragile ⇒ Make update atomic
& Integrate reconfiguration

reUpNix contribution

NixOS

/etc

systemd

sshd

/etc

NixOS

nix/store

reconfiguration
Differential Update Transfer

Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd')...
"diff(C, C') by sender (knows C & C')
⇒ unidirectional transfer
Differential Update Transfer

Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd') ...
“diff(C, C’) by sender (knows C & C’)
⇒ unidirectional transfer
Differential Update Transfer

Many/all new components, but very similar component exists. Find "before" component, update differentially. No clear package name/ID, or lineage. ⇒ walk pair of dependency graphs. Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...

"diff(C, C')" by sender (knows C & C') ⇒ unidirectional transfer.
Differential Update Transfer

Many/all new components, but very similar component exists

Find “before” component, update differentially

No clear package name/ID, or lineage

⇒ walk pair of dependency graphs

Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...

“diff(C, C’) by sender (knows C & C’)

⇒ unidirectional transfer
Differential Update Transfer

Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage ⇒ walk pair of dependency graphs
Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...
"diff(C, C') by sender (knows C & C') ⇒ unidirectional transfer
Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...
“diff(C, C’) by sender (knows C & C’)
⇒ unidirectional transfer
Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd') ...
"diff(C, C') by sender (knows C & C')
⇒ unidirectional transfer
Differential Update Transfer

Many/all new components, but very similar component exists

Find “before” component, update differentially

No clear package name/ID, or lineage

⇒ walk pair of dependency graphs

Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...

“diff(C, C’) by sender (knows C & C’)

⇒ unidirectional transfer

TUHH
Many/all new components, but very similar component exists
Many/all new components, but very similar component exists

Find “before” component, update differentially
Many/all new components, but very similar component exists

Find “before” component, update differentially

No clear package name/ID, or lineage
Many/all new components, but very similar component exists

Find “before” component, update differentially

No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer:
Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer: \text{diff}(\text{NixOS}, \text{NixOS}'),
Many/all new components, but very similar component exists
Find “before” component, update differentially
No clear package name/ID, or lineage
⇒ walk pair of dependency graphs
Transfer: \( \text{diff(NixOS, NixOS')} \), \( \text{diff(sshd, sshd')} \),
Differential Update Transfer

- Many/all new components, but very similar component exists
- Find “before” component, update differentially
- No clear package name/ID, or lineage
  ⇒ walk pair of dependency graphs
- Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'),
  diff(glibc, glibc'),...
Differential Update Transfer

- Many/all new components, but very similar component exists
- Find “before” component, update differentially
- No clear package name/ID, or lineage
  ⇒ walk pair of dependency graphs
- Transfer: \( \text{diff(NixOS, NixOS')}, \text{diff(sshd, sshd')}, \text{diff(openssl, openssl')}, \text{diff(glibc, glibc')} \)
Many/all new components, but very similar component exists

Find “before” component, update differentially

No clear package name/ID, or lineage
⇒ walk pair of dependency graphs

Transfer: \text{diff}(\text{NixOS, NixOS'}, \text{diff}(\text{sshd, sshd'}), \text{diff}(\text{openssl, openssl'}), \text{diff}(\text{glibc, glibc'}), \text{diff}(\text{systemd, systemd'}),
Many/all new components, but very similar component exists

Find “before” component, update differentially

No clear package name/ID, or lineage
⇒ walk pair of dependency graphs

Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...
Differential Update Transfer

- Many/all new components, but very similar component exists
- Find “before” component, update differentially
- No clear package name/ID, or lineage ⇒ walk pair of dependency graphs
- Transfer: diff(NixOS, NixOS'), diff(sshd, sshd'), diff(openssl, openssl'), diff(glibc, glibc'), diff(systemd, systemd'), ...
- “diff(C, C’)” by sender (knows C & C’) ⇒ unidirectional transfer
**Differential Update Transfer: Results**

\[ \text{diff}(C, C') = \]

- **File Deduplication**
- **FD + Reference chunking**
- **FD + fixed 64 byte chunking**
- **FD + R + 64**

### Base System
- **Base System**: 139.3 MiB
- **Update Libc**: 141.4 MiB

### Base w/o Kernel
- **Base w/o Kernel**: 107.8 MiB
- **Update Libc**: 109.8 MiB

### 75 Days
- **Base System**: 36.1 MiB
- **Update Libc**: 30.5 MiB

**REUPNix: Reconfigurable and Updateable Embedded Systems**
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
- **System**: multiple systems on one reliable FS (e.g., journaling)
- **Boot**: FAT32 for firmware compatibility

**Flash Disk**

<table>
<thead>
<tr>
<th>Boot (FAT32)</th>
<th>System (ro)</th>
<th>Data (rw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>kernel + initrd</td>
<td>• before update • after update</td>
<td>User data, Logs, ...</td>
</tr>
</tbody>
</table>

⇒ applicable to other operating systems!
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
- **System**: multiple systems on one reliable FS (e.g., journaling)
- **Boot**: FAT32 for firmware compatibility

Flash Disk

<table>
<thead>
<tr>
<th>Boot (FAT32)</th>
<th>System (ro)</th>
<th>Data (rw)</th>
</tr>
</thead>
</table>
| entry before | • before update  
• after update | User data, Logs, ... |
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
- **System**: multiple systems on one reliable FS (e.g., journaling)
- **Boot**: FAT32 for firmware compatibility

Flash Disk

<table>
<thead>
<tr>
<th>Boot (FAT32, rw)</th>
<th>System (ro)</th>
<th>Data (rw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry before</td>
<td>before update</td>
<td>User data, Logs, ...</td>
</tr>
<tr>
<td>entry after</td>
<td>after update</td>
<td></td>
</tr>
</tbody>
</table>

⇒ applicable to other operating systems!
Atomic Bootloader Updates

- **Goal:** safe/atomic updates for NixOS on any hardware/firmware
- **System:** multiple systems on one reliable FS (e.g., journaling)
- **Boot:** FAT32 for firmware compatibility **but unreliable**

Flash Disk

<table>
<thead>
<tr>
<th>Boot (FAT32,rw)</th>
<th>System (ro)</th>
<th>Data (rw)</th>
</tr>
</thead>
<tbody>
<tr>
<td>entry before</td>
<td>● before update</td>
<td>User data, Logs, ...</td>
</tr>
<tr>
<td>entry after</td>
<td>● after update</td>
<td></td>
</tr>
</tbody>
</table>
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
- **System**: multiple systems on one reliable FS (e.g., journaling)
- **Boot**: FAT32 for firmware compatibility but unreliable
- **Solution**: A/B partitioning for Boot

Flash Disk

<table>
<thead>
<tr>
<th>Table A</th>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table B</td>
<td>boot-1</td>
<td>boot-2</td>
<td>system</td>
<td>data</td>
</tr>
</tbody>
</table>

Table 2

| boot-2 | boot-1 | system | data |

Flash Disk

- Boot-A (FAT32,ro)
  - entry before
- Boot-B (FAT32,ro)
  - entry after
- System (ro)
  - before update
  - after update
- Data (rw)
  - User data,
    Logs,. . .
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
- **System**: multiple systems on one reliable FS (e.g., journaling)
- **Boot**: FAT32 for firmware compatibility but unreliable
- **Solution**: A/B partitioning for Boot

Flash Disk

**Table A**

<table>
<thead>
<tr>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
</table>

**Table B**

<table>
<thead>
<tr>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
</table>

**Table 2**

<table>
<thead>
<tr>
<th>boot-2</th>
<th>boot-1</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
</table>

**Boot (FAT32)**

- kernel + initrd

**Boot (FAT32, rw)**

- entry before

**Boot-A (FAT32, ro)**

- entry before

**Boot-B (FAT32, ro)**

- entry after

**System (ro)**

- before update
- after update

**Data (rw)**

User data, Logs, ...

⇒ applicable to other operating systems!
Atomic Bootloader Updates

- **Goal**: safe/atomic updates for NixOS on any hardware/firmware
- **System**: multiple systems on one reliable FS (e.g., journaling)
- **Boot**: FAT32 for firmware compatibility but unreliable
- **Solution**: A/B partitioning for Boot switched via GPT

Flash Disk

<table>
<thead>
<tr>
<th>Table A</th>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table B</td>
<td>boot-1</td>
<td>boot-2</td>
<td>system</td>
<td>data</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Boot (FAT32)</th>
<th>kernel + initrd</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boot (FAT32)</td>
<td>entry before</td>
</tr>
<tr>
<td>Boot-A (FAT32, ro)</td>
<td>entry before</td>
</tr>
<tr>
<td>Boot-B (FAT32, rw)</td>
<td>entry after</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>System (ro)</th>
<th>before update</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data (rw)</td>
<td>after update</td>
</tr>
</tbody>
</table>

| Data (rw) | User data, Logs, ... |

⇒ applicable to other operating systems!
Atomic Bootloader Updates

- **Goal:** safe/atomic updates for NixOS on any hardware/firmware
- **System:** multiple systems on one reliable FS (e.g., journaling)
- **Boot:** FAT32 for firmware compatibility but unreliable
- **Solution:** A/B partitioning for Boot switched via GPT

Flash Disk

Table A

<table>
<thead>
<tr>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
</table>

Table B

<table>
<thead>
<tr>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
</table>

Table 2

<table>
<thead>
<tr>
<th>boot-1</th>
<th>boot-2</th>
<th>system</th>
<th>data</th>
</tr>
</thead>
</table>

Boot-A (FAT32,rw)

- entry before

Boot-B (FAT32,ro)

- entry after

System (ro)

- before update
- after update

Data (rw)

- User data, Logs, ...

⇒ applicable to other operating systems!
Atomic Bootloader Updates

Goal: safe/atomic updates for NixOS on any hardware/firmware

System: multiple systems on one reliable FS (e.g., journaling)

Boot: FAT32 for firmware compatibility but unreliable

Solution: A/B partitioning for Boot switched via GPT

Flash Disk

⇒ applicable to other operating systems!
Change system’s executed software configuration
Change system’s executed software configuration

Update: 1 → 2 → 3
Change system’s executed software configuration

Update: 1 → 2 → 3

Reconfigure:

A → B → C

Correctness: independent of previous profile
Complete: e.g. change kernel
Change system’s executed software configuration:

Update: \[1 \rightarrow 2 \rightarrow 3\]  
Reconfigure: \[A \circlearrowleft B \circlearrowleft C \circlearrowleft\]
Reconfiguration

- Change system’s executed software configuration
  - Update: 1 → 2 → 3
  - Reconfigure: A, B, C

- NixOS: multiple variants ("profiles")
Reconfiguration

- Change system's executed software configuration
  
  Update: 1 → 2 → 3
  
  Reconfigure: A → B → C
  
- NixOS: multiple variants ("profiles")
Change system’s executed software configuration

Update: 1 → 2 → 3

Reconfigure:

NixOS: multiple variants (“profiles”)
Reconfiguration

- Change system’s executed software configuration
  
  Update: 1 \rightarrow 2 \rightarrow 3  
  Reconfigure:  

- NixOS: multiple variants ("profiles")
- reUpNix: reconfig \Rightarrow \text{reboot}
Reconfiguration

- Change system’s executed software configuration
  
  Update: 1 → 2 → 3
  
  Reconfigure: A → B → C

- NixOS: multiple variants ("profiles")
- reUpNix: reconfig ⇒ reboot
  
  Correctness: independent of previous profile
Change system’s executed software configuration

- Update: 1 → 2 → 3
- Reconfigure: A → B → C
- Default

- NixOS: multiple variants ("profiles")
- reUpNix: reconfig ⇒ reboot
  - Correctness: independent of previous profile
  - Completeness: e.g. change kernel
Change system’s executed software configuration

Update: 1 → 2 → 3

Reconfigure: A → B → C

- NixOS: multiple variants (“profiles”)
- reUpNix: reconfig ⇒ reboot
  - Correctness: independent of previous profile
  - Completeness: e.g. change kernel
Change system’s executed software configuration

Update: 1 → 2 → 3

Reconfigure: A → B → C

NixOS: multiple variants ("profiles")

reUpNix: reconfig ⇒ reboot
- Correctness: independent of previous profile
- Completeness: e.g. change kernel

<table>
<thead>
<tr>
<th>PowerOff</th>
<th>RPi Firmw.</th>
<th>U-Boot</th>
<th>Linux</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.9</td>
<td>7.9</td>
<td>4.2</td>
<td>10.8</td>
</tr>
</tbody>
</table>

Boot Time [s]

| 0 | 1.9 | 9.78 | 14.01 | 24.79 |

REUPNix: Reconfigurable and Updateable Embedded Systems
NixOS: flexible, reproducible, predictable, reconfigurable
- Not suited for embedded

Solved NixOS problems for embedded
- Reduced system size (~1.1 GiB → 155 MiB)
- Updates up to 99.88 % smaller (mitigate change amplification)
- Atomic bootloader config update (switch A/B part via GPT)

reUpNix key features
- Reconfiguration / multi-system setup
- Efficient OCI services (w/o Docker, file-deduplicated)
- Unidirectional communication update transfers
The End
Nix Store Reference Pinning

```
/nix/store
fd53d:
rx2gn:
d3lzb:
```

Diagram showing the references to `fd53d`, `rx2gn`, and `d3lzb`. The diagram illustrates how these references are connected within the Nix store.
Update Deduplication

(a) File Dedup.  (b) Blocks  (c) Refs  (d) Ref+Block

O N  O N  O N  O N

Nix Ref.  transfer!  reuse!