State of Stateless

A Talk About Immutability in Debian

Arun Mani J & Ragul R

A Definition of Stateless

State

- It is the information present on the system at any given moment.
- For an OS, state is the files or in a higher-level - the packages and their configuration.
- Examples include our conventional operating systems like Debian, Ubuntu etc.

Stateless

- Stateless OS can be deployed and run such that the host OS and the applications running in it are completely independent of each other.
- In such systems, the root directory is often read-only.
- There will be a selected list of directories, mostly, `/var`, `/etc`, `/tmp` to write data.

Ecosystem of Stateless

Immutability

- An immutable OS doesn't allow any changes.
- Either the OS can make its file system completely read-only.
- Or the changes made is lost on reboot.

Declarative

- It is the recipe to make the system we need.
- It contains the list of packages, their version and preferably checksum.
- It can used by anyone to create the same environment.

Reproducible

- If a declaration can be used to create bit-bit equivalent systems, then the declaration is 100% reproducible.
- It allows us to verify whether a different source code was used to build a software.

Isolation

- It tells us whether an app A can modify an other app B without B's knowledge.
- Sometimes, the isolation is kept only between host and the user.

Self Contained

- It is a measure of how much an app depends on the host for its dependencies.
- Impossible to get 100% but combined with declarative and 100% reproducible environment, we can attain an optimal value.

Snapshot

- It is the image of a file system at a particular moment in time.
- Ideally, snapshot should be made whenever the file system is modified.
- Snapshots allow us to revert back destructive changes.

Atomicity

- The principle of atomicity states that an update is either successfully complete or an utter failure.
- It means, the system is never left in a broken state, no matter what catastrophe happens.

Issues in Stateless

Security

- Stateless allows `$HOME`, `/tmp` etc. to be read-write.
- Nothing prevents a malicious or buggy app from stealing data like SSH keys.
- Hence, no security, unless otherwise configured.

Disk Usage

- Frequent changes results in eating up of disk space by snapshots.
- Reproducibility might have to build every dependency from source. This takes up a good amount of space.
- However, sharing of libraries can help.

Bandwidth

- Downloading dependencies can consume a lot of bandwidth.
- On production systems, with same setup in each system, a local mirror can be used.

Build Time

- Building every dependency delays setup.
- A centralized build system can be used to cache and speed-up the process.

Usage Perspective

Developer

- A declarative system allows us to pin-point the exact version of every single package required.
- With 100% reproducibility, we can ensure that every developer has the same development environment.
- This ideally solves the “it works on my computer” problem.

System Administrator

- A declarative configuration allows us to deploy an application in an automated manner.
- Vulnerabilities can be tracked for every dependency easily.
- Replication of a setup across multiple system gets more convenient.
- Snapshots lets us rollback buggy configurations with less hassle.

Normal User

- Atomicity and snapshots are useful features in home machines where crashes in terms of hardware and power is unavoidable.
- Declarative approach allows easy installation with less manual intervention.
- Users can match the exact requirements of an application, which solves the issue of “library version mismatch”.

Inspiration & Existing Technology

Libostree

- Formerly known as OSTree.
- It is a system for versioning updates.
- It is Git for operating system binaries.
- `libostree` is a library which can be used the underlying host to version and deploy the changes.
- It is used by endless OS, Flatpak, Fedora's immutable spins etc.

Libostree – Overview

- Create a new repository.
- Make changes to the OS.
- Use OSTree to commit it.
- Deploy the new commit.
- Revert back if needed.

Libostree – Getting Started

- Install `ostree`

  ```
  $ sudo apt install ostree
  ```

- Initialize a repository `repo` in current working directory.

  ```
  $ ostree init --repo=repo
  ```

- Create work tree.

  ```
  $ mkdir tree
  ```

Libostree – Getting Started

- Make some changes.

```
$ mkdir tree/server
```

```
$ echo “python3 -m http.server” > tree/server/app.sh
```

- Commit to branch `main`.

```
$ ostree --repo=repo commit --branch=main --subject="Created server" tree/
```

```
5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
```

Libostree – Getting Started

- Make changes to changes.

```
$ printf "echo 'Running'\npython3 -m
http.server\n" > tree/server/app.sh
```

- Commit the changes.

```
$ ostree --repo=repo commit --branch=main --
subject="Added logging" tree/

9b75290f6a6359a2a3471022cbba4b724e45105b313ae8f6
c103a2f79e82a857
```

Libostree – Getting Started

- **Show the log of commits.**

```
commit 9b75290f6a6359a2a3471022cbba4b724e45105b313ae8f6c103a2f79e82a857
Parent: 5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
ContentChecksum:
4ad28e4a6461bd64b920f72f86c0d16edc544c4a1f26060518ebb900025d496a
Date: 2023-09-07 07:16:57 +0000
    Added logging
commit 5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
ContentChecksum:
e1aef671f29c63c748142b89d3657ad8a28f3ceffdd545c08dc2f4479aa4ac7b
Date: 2023-09-07 07:15:20 +0000
    Created server
```

Libostree – Getting Started

- List the files in `main`.

```
$ ostree --repo=repo ls main
d00755 1000 1000      0 /
d00755 1000 1000      0 /server
```

Libostree – Getting Started

- View the files in `main` or a commit.

```
$ ostree --repo=repo cat main server/app.sh
echo 'Running'
python3 -m http.server
$ ostree --repo=repo cat 9b752 server/app.sh
echo 'Running'
python3 -m http.server
$ ostree --repo=repo cat 5891b server/app.sh
python3 -m http.server
```

Libostree – Getting Started

- Make some errors.

```
$ printf "echo 'Running'\npython3 -m
http.server\n" > tree/server/app.sh
```

```
$ ostree --repo=repo commit --branch=main --
subject="Updated to FTP" tree/
```

```
39f786f06974701a78fe2888b843cdf653c1f9f730600fb5
d2409594d52ae791
```

Libostree – Getting Started

- Check the logs.

```
$ ostree --repo=repo log main
commit 39f786f06974701a78fe2888b843cdf653c1f9f730600fb5d2409594d52ae791
Parent: 9b75290f6a6359a2a3471022cbba4b724e45105b313ae8f6c103a2f79e82a857
ContentChecksum: ccde54526baba3c41591202a27b3d37001bcbdffa03cfbe5214e4685c70ad869
Date: 2023-09-07 07:18:15 +0000
    Updated to FTP
commit 9b75290f6a6359a2a3471022cbba4b724e45105b313ae8f6c103a2f79e82a857
Parent: 5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
ContentChecksum: 4ad28e4a6461bd64b920f72f86c0d16edc544c4a1f26060518ebb900025d496a
Date: 2023-09-07 07:16:57 +0000
    Added logging
commit 5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
ContentChecksum: e1aef671f29c63c748142b89d3657ad8a28f3ceffdd545c08dc2f4479aa4ac7b
Date: 2023-09-07 07:15:20 +0000
    Created server
```

Libostree – Getting Started

- **Rollback to a commit.**

```
$ ostree --repo=repo reset main 9b752
$ ostree --repo=repo log main
commit 9b75290f6a6359a2a3471022cbba4b724e45105b313ae8f6c103a2f79e82a857
Parent: 5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
ContentChecksum: 4ad28e4a6461bd64b920f72f86c0d16edc544c4a1f26060518ebb900025d496a
Date: 2023-09-07 07:16:57 +0000
    Added logging
commit 5891b5b522d5df086d0ff0b110fbd9d21bb4fc7163af34d08286a2e846f6be03
ContentChecksum: e1aef671f29c63c748142b89d3657ad8a28f3ceffdd545c08dc2f4479aa4ac7b
Date: 2023-09-07 07:15:20 +0000
    Created server
```

Libostree – Getting Started

- Deploy the rollback

```
$ cat tree/server/app.sh # Before
echo 'Running'
python3 -m ftp.server
$ ostree --repo=repo checkout --union 9b752 tree/
$ cat tree/server/app.sh # After
echo 'Running'
python3 -m http.server
```

Libostree – Getting Started

- Pack a commit into an archive.

```
$ ostree --repo=repo export main > main.tar.gz
$ tar --list --file=main.tar.gz
./
server/
server/app.sh
```

Libostree – Merits

- It is designed for the purpose of immutability.
- Similarity to Git, making it easy to adapt.
- Contains built-in functions to help in majority of the use cases.

Libostree - Demerits

- Not meant for direct end-user usage.
- Lack of much documentation or help outside the official ones.
- Needs non-trivial changes for usage.

Fedora Silverblue (`rpm-ostree`)

- rpm-ostree is a hybrid image and package system.
- It combines libostree as a base image format, and uses RPM for packages.
- Fedora Silverblue is an immutable variant of Fedora Workstation using `rpm-ostree`.

Fedora Silverblue – Overview

- Fedora Silverblue supports three ways of installing software.
- Flatpak – For GUI apps.
- Toolbox – For CLI and development apps.
- Package layering – For core-level packages like drivers etc.
- Each update of the OS creates an entry in boot-loader.

Fedora Silverblue – Getting Started

- Update the OS.

  ```
  $ rpm-ostree upgrade
  ```

- Temporary rollback.

  Boot the previous version from boot-loader menu.

- Permanent rollback to previous version.

  ```
  $ rpm-ostree rollback
  ```

Fedora Silverblue – Merits

- Complete abstraction from libostree.
- Easy to get started and use thanks to GUI and simple commands.
- All the other advantages of libostree.

Fedora Silverblue – Demerits

- Setting up of development environment can be difficult.
- Packages needs to modified to work with the new restrictions.

Nix

- Nix is a purely functional package manager
- It helps you make sure that package dependency specifications are complete.
- When installing a package, Nix calculates a unique hash to store it in `/nix/store`.
- The risk of incomplete dependencies are greatly reduced.

Nix – Installation

- Multi-user installation

```
$ sh < (curl -L https://nixos.org/nix/install)
--daemon
```

- Single user installation

```
$ sh < (curl -L https://nixos.org/nix/install)
--no-daemon
```

Nix – Overview

- Prepare a declaration.
- Spin up the environment
- Update the declaration.

Nix – Install a Package

- Invoking `nix-env`.

```
$ nix-env --install python3
```

- Check the installed package.

```
$ command -V python3
/nix/store/jhflvwr40xbb0xr6jx4311icp9cym1fp-python3-3.10.12/bin/python3.10
```

Nix – Uninstall a Package

- Query the installed packages.

```
$ nix-env --query
python3
```

- Uninstall `python3`.

```
$ nix-env --uninstall python3
```

Nix – Switch Generations

- Install another package.

```
nix-env --install tree
```

- List the generations.

```
$ nix-env --list-generations
1   2023-09-05 22:55:58
2   2023-09-05 23:02:11 (current)
```

- Switch to the previous generation.

```
$ nix-env –switch-generation=1
$ tree
command not found: tree
```

Nix – Ad-hoc Environment

- Make a temporary development environment.

```
$ nix-shell -p python3
```

- Nix creates an isolated environment where declared packages are available.

```
$ python3 -c "print('hello')"
hello
```

Nix – Reproducible Environment

```
$ cat shell.nix
{ pkgs ? import (fetchTarball
"https://github.com/NixOS/nixpkgs/archive/06278c77
b5d162e62df170fec307e83f1812d94b.tar.gz") {}}:
pkgs.mkShell {
    packages = [
    (pkgs.python3.withPackages (ps: [ps.flask]))
    pkgs.curl
];}
```

GNU Guix

- Guix implements the functional package management discipline pioneered by Nix.
- Advantage over Nix is that built packages can be used in the environment where Guix is not installed.
- But Guix requires knowledge about Scheme to write package definitions

Guix – Getting started

- Installation

```
$ sudo apt install guix
```

- Create an Ad-hoc development environment

```
$ guix shell python3
$ command -v python
/gnu/store/66qalq2h24ax12vp059fdjjahcmqp1pz-
python-3.10.7/bin/python3
```

Guix – Ad-hoc Environment

- Lets create an Ad-hoc Environment

```
$ guix shell python python-numpy
$ python
>>> import numpy as np
>>> np.__version__
'1.24.2'
```

Guix – Packaging application

- Packages can be built to use in the environment where guix is not available.
- Invoking guix pack

```
$ guix pack hello
$ guix pack -RR -S /mybin/hello=bin hello
$ ./mybin/hello
Hello, world!
```

Guix – Package a deb archive

- Guix have builtin support for packaging the application in deb and rpm format.
- To produce a Debian archive containing all the specified binaries and symbolic links, that can be installed on top of any dpkg-based GNU(/Linux) distribution.

```
$ guix pack -f deb -C xz -S /usr/bin/hello=bin/hello hello
$ sudo dpkg -i bpbpflc42jwryfrjpkqix3vnm8cdbmnr-hello-deb-pack.deb
$ hello
Hello, World!
```

But chroot

chroot

- Is not a solution.
- For this to work, we have to make a minimal environment with at least `bash`, `apt` etc.
- It is a tedious task, though `debootstrap` can help.
- But versioning, rollback has to be manually implemented.

Thought Experiment

How An Immutable Debian Could Be Like?

A Different `apt`

- `apt` could support installation of packages to a user's specific directory.

  ```
  $ apt install --dir=foo python3
  ```

- Combined with `direnv`, these packages are loaded only inside the directory.
- This can give an ad-hoc environment to test around packages without polluting global packages.

Snapshots in `apt`

- Every change in packages can result in a new snapshot.
- These snapshots could be added to boot-loader, so users can move to a previous one if needed.
- However, `libostree` like library might be needed at some point for compression, differential storage etc.

Allow Us To Introduce Ourselves

BTRFS, Flatpak, Containers and Co.

BTRFS

- Btrfs is a file system based on the copy-on-write (COW) principle with a logical volume manager.
- It has support for subvolumes with different properties like permissions and quota.
- Snapshots can be made of subvolumes.
- CoW ensures that we can rollback changes made to a file.

Flatpak

- Built on top of `libostree`.
- Applications that are installed is stored in a local version control repository, and is then mapped into the local file system.
- Applications runs in a sandbox without affecting the host environment.

Containers

- Containers made through Docker, Podman etc. create a light-weight virtual machines.
- They can be used to create environments that work the same irrespective of the host.
- Containers allow commit, which is similar to pausing and resuming it any required instant.

How Do I Start?

Development

- We can use Nix and GNU Guix for isolated development environments, which solves many dependency problems.
- Nix and GNU Guix allow reproducible environments that makes sure everyone gets the same packages.
- Declarative package management lets easy tracking of versions.

Deployment

- Apps can be deployed as containers.
- This way we keep the packages installed on Debian minimal.
- This way we ensure that the dependencies for every app is self-contained.

Conclusion

- This talk worked on our computer.
- No Debian packages were hurt in making this presentation.


THANK YOU