



ELISA
Enabling **Linux** in
Safety Applications



WORKSHOP

LUND 2024

A primer on Nix

and its relevance for safety-critical software systems



Nix As a Package Manager



ELISA
Enabling **Linux** in
Safety Applications



WORKSHOP



Nix as Database



Nix as a Configuration Language



Nix as a Linux Distribution



What is Nix? - Components

- **Nix CLI**
 - For building, installing, and managing packages.
- **Nix Store**
 - Stores each package in isolation
- **Nix Language**
 - A purely functional declarative language
- **Nixpkgs**
 - Over 100,000 packages.
- **NixOS**
 - Linux distribution built on top of Nix; taking declarativity to the next level

Core concepts: Nix Store - a Database of packages

```
dr-xr-xr-x - root 1 jan 1970 0a0rfnlqi7287a8319isbqiaqk4y9j0v-source
dr-xr-xr-x - root 1 jan 1970 0a6c52f5kdvkdggkz0d97yiaa05r1lbz-bat-0.24.0-fish-completions
.r--r--r-- 3,5k root 1 jan 1970 0a7drakv7wss56dbglza593glia9ymdd-source.drv
.r--r--r-- 2,9k root 1 jan 1970 0a92vy7ncihx1kcw0cncdj8m653yxlis-Babel-2.14.0.tar.gz.drv
.r--r--r-- 4,7k root 1 jan 1970 0a2444q1gpvyy8xwnx88w0b7x72pxy3r-perl-5.38.2.drv
dr-xr-xr-x - root 1 jan 1970 0aavdx9m5ms1cj5pb1dx0brbrbigy8ij-source
.r--r--r-- 529 root 1 jan 1970 0abw31f38k2dj441iwr0lfz2c4ljc8jj-usermod.pam
.r--r--r-- 913 root 1 jan 1970 0af36pnpmn76j26vq83p7bfb6ks0kc9f-sshd.conf-final
dr-xr-xr-x - root 1 jan 1970 0afc5w03x63bwas2xfl8vlmvhciky1h8-expand-response-params
.r--r--r-- 3,8k root 1 jan 1970 0aghw59j6mp12ybdc49rlwc2iqnmxlbg-libclc-17.0.6.drv
```

Core concepts: Nix Store - a Database of packages

```
> tree /nix/store/g1n2vryg06amvcc1avb2mcq36faly0mh-hello-2.12.1
/nix/store/g1n2vryg06amvcc1avb2mcq36faly0mh-hello-2.12.1
├── bin
│   └── hello
├── share
│   ├── info
│   │   └── hello.info
│   ├── locale
│   │   └── ast
│   │       └── LC_MESSAGES
│   │           └── hello.md
│   └── ...
└── ...
```

Core concepts: Expressions

```
stdenv.mkDerivation {
  name = "hello";
  src = ./src;
  buildInputs = [ libyaml ];
  nativeBuildInputs = [ pkg-config ];

  configurePhase = ''
    declare -xp
  '';

  buildPhase = ''
    $CC "$src/hello.c" -o ./hello
  '';

  installPhase = ''
    mkdir -p "$out/bin"
    cp ./hello "$out/bin/"
  '';
}
```

Core concepts: Derivations

```
{
  "derivation": {
    "name": "hello-19700101",
    "builder": "/nix/store/dzrvibwj2vjwqmc34wk3x1ffsjpp4av7-bash-4.4-p23/bin/bash",
    "env": {
      "buildInputs": "/nix/store/5r6lgv5il4068nzwinb6nwnx9wgirfxs-libyaml-0.2.5",
      "nativeBuildInputs": "/nix/store/rjxjv5rdh4158kzaxks93i0lkddihld-pkg-config-wrapper-0.29.2",
      "src": "/nix/store/p57wrsd3904i1kqmd3pxhs6krvi2i4hp-j3j3gwg3kavfjbi2bx7p03z57w2zahyz-source",
      "stdenv": "/nix/store/14c13y7lg30gw97gphfd7hg8x0cq2czk-stdenv-linux",
    },
    "inputSrcs": [
      "/nix/store/p57wrsd3904i1kqmd3pxhs6krvi2i4hp-j3j3gwg3kavfjbi2bx7p03z57w2zahyz-source"
    ],
    "outputs": {
      "out": {
        "path": "/nix/store/rgkvza100f8wd0alqkrjz85b6ahnjycv-hello-19700101"
      }
    }
  }
}*
```

* Abbreviated for convenience

How Nix Solves the Challenges from Traditional Package Management



What is Nix? - Key features

- **Declarative**
 - Definition: Describe what the system or package should look like, not how to build it.
 - Benefit: Simplifies configuration management by defining the desired state, making it easier to manage complex systems consistently.
- **Reproducible**
 - Definition: Ensures that builds is performed in the exact same environment.
 - Benefit: Guarantees that software behaves the same way in development, testing, and production, eliminating "it works on my machine" issues.
- **Reliable**
 - Definition: Each build is isolated and has no side-effects on the system, with strong guarantees about dependencies.
 - Benefit: Increases system stability and predictability, reducing downtime and maintenance effort.

Challenge: Dependency Hell

- Issue: Managing complex dependencies and version conflicts can lead to unstable systems and difficult debugging.
- Solution with Nix: Nix's purely functional package management ensures that all dependencies are explicitly declared and isolated. This prevents version conflicts and allows multiple versions of the same package to coexist without issues.

Challenge: Development Environments

- Issue: Traditional Development Environments rely on non-reproducible setups, causing the infamous “but it works on my machine” arguments.
- Solution: Nix’s reproducibility ensures every developer has the exact same setup to the individual bit.
- Demo time!

Demo: Backup Video



```
me@desktop-0906040:~$ sudo ./build
Compiling prog-macro2 v1.8.19
Compiling serde v1.0.197
Compiling libc v0.2.153
Compiling lock_api v0.4.10
Compiling parking_lot_core v0.9.6
Compiling atomic v0.8.1
Compiling serde_json v2.0.114
Compiling openssl-sys v0.9.80
Compiling zerocopy v0.7.32
Compiling bitflags v1.3.2
Compiling rpsa-autocerts v0.4.0
Compiling equivalent v0.1.1
error: linker `x86_64-linux-gnu-gcc` not found
  |
  | note: No such file or directory (as error 3)
  |
error: could not compile `serde_json` (build script) due to 1 previous error
warning: build failed, waiting for other jobs to finish...
error: could not compile `lock_api` (build script) due to 1 previous error
error: could not compile `ashash` (build script) due to 1 previous error
error: could not compile `serde` (build script) due to 1 previous error
error: could not compile `parking_lot_core` (build script) due to 1 previous error
error: could not compile `prog-macro2` (build script) due to 1 previous error
error: could not compile `openssl-sys` (build script) due to 1 previous error
error: could not compile `libc` (build script) due to 1 previous error
error: could not compile `libc` (build script) due to 1 previous error
me@desktop-0906040:~$
```

Challenge: Traceability I

```
{
  "derivation": {
    "name": "hello-19700101",
    "builder": "/nix/store/dzrvibwj2vjwqmc34wk3x1ffsjpp4av7-bash-4.4-p23/bin/bash",
    "env": {
      "buildInputs": "/nix/store/5r6lgv5il4068nzwinb6nwnx9wgirfxs-libyaml-0.2.5",
      "nativeBuildInputs": "/nix/store/5f6aw7kg4lfb3rswcfhml94fr8fd4495-hook /nix/store/rjxjv5rdh4158kzaxks93i01kddihld-pkg-config-wrapper-0.29.2",
      "src": "/nix/store/p57wrsd3904iilkqmd3pxhs6krvi2i4hp-j3j3gwg3kavfjbi2bx7p03z57w2zahyz-source",
      "stdenv": "/nix/store/14cl3y7lg30gw97gphfd7hg8x0cq2czk-stdenv-linux",
    },
    "inputSrcs": [
      "/nix/store/p57wrsd3904iilkqmd3pxhs6krvi2i4hp-j3j3gwg3kavfjbi2bx7p03z57w2zahyz-source"
    ],
    "outputs": {
      "out": {
        "path": "/nix/store/rgkvza100f8wd0alqkrjz85b6ahnjycv-hello-19700101"
      }
    }
  }
}*
```

* Abbreviated for convenience

Challenge: Traceability II - Demo

```
v@kali:~$ cyclonede
cyclonede on F with [[]] via C
] cyclonede --help
Converts BSI-encoding tool's output into Cyclonede.

Usage: cyclonede [OPTIONS] [INPUT_FILE]

Options:
  -f, --file <FILE>          Path to the input BSI/STIX file
  -o, --output-file <OUTPUT_FILE>  Optional path to the output Cyclonede file (default: stdout)
  --backend <BACKEND>        Backend to use for BSI evaluation/encoding (default: workbench) [possible values:
  workbench]
  --include-usr-info          Attempt to include userfile data in the log
  --opt-usr-only              Attempt to only include runtime dependencies in the log
  --usr-spec <SPEC>          Optional user specification to output (default: cyclonede.i.d.json) [possible values:
  cyclonede.i.d.json, cyclonede.i.d.xml, cyclonede.i.d.json, cyclonede.i.d.xml]
  -v, --verbose...           Increase logging verbosity
  -q, --quiet...             Decrease logging verbosity
  -h, --help                 Print help
  -V, --version              Print version

v@kali:~$ cyclonede --help
```

Licensing of Workshop Results

All work created during the workshop is licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) [<https://creativecommons.org/licenses/by/4.0/>] by default, or under another suitable open-source license, e.g., GPL-2.0 for kernel code contributions.

You are free to:

- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.