

arik-grahl.de/talks/nixcon-2026

Nix for Cloud Native Platform Engineering

From Imperative Templating to Functional Derivation

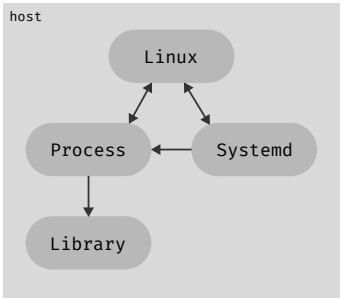
echo $(whoami)



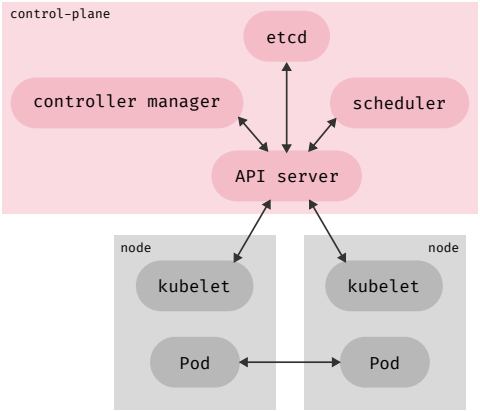
```
{ ... }:
{
  person = {
    meta = with pronouns; {
      name = { firstName = "Arik"; surname = "Grahl"; };
      pronouns = heHim;
    };
    work = with roles; {
      company = "SysEleven";
      role = staffSystemsArchitect;
    };
    contact = {
      web = "www.arik-grahl.de";
      gitlab = "gitlab.com/arik.grahl";
      github = "github.com/arikgrahl";
      linkedin = "linkedin.com/in/arikgrahl";
      mastodon = "chaos.social/@arikgrahl";
      matrix = "@arik:matrix.arik-grahl.de";
    };
  };
}
```

NixOS vs. Kubernetes

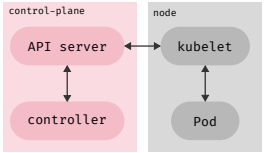
Nixos



Kubernetes



Cloud Native









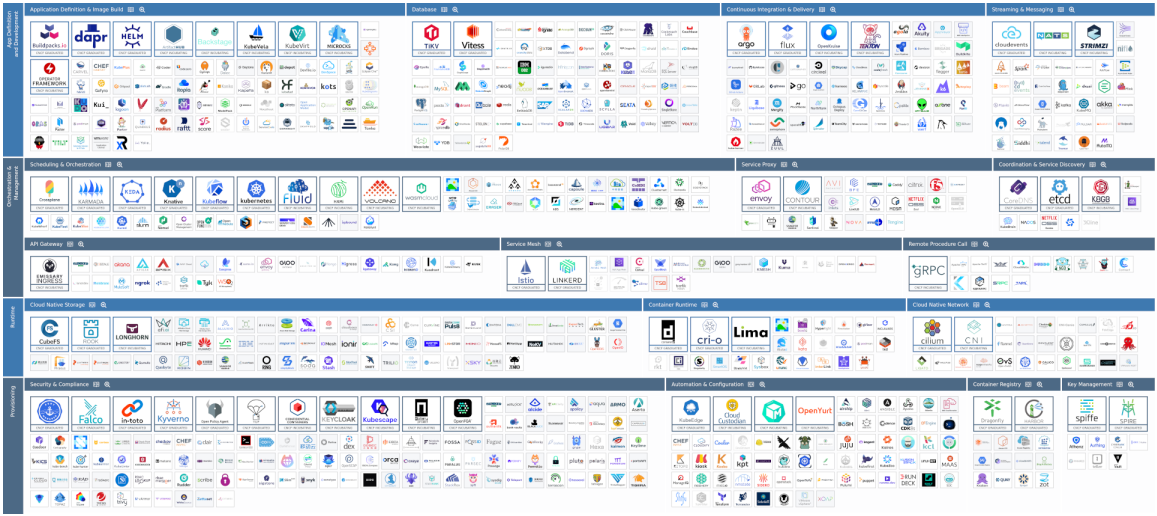


kro

- the Good: Kubernetes API
  - reconciliation towards declared state
  - reasonable abstraction
  - minimal (cloud) vendor lock-in
- the Bad: Controlling complexity
  - stacking applications without abstraction
  - opaque combination and reconcilation logic in CI/CD pipelines
  - inventory and self-service of ever growing resources
- the Ugly: Packaging by YAML modification
  - Kustomize as a imperative mutation
  - Helm as a YAML templating engine
  - KRO as a schema-based but unversioned generator

Cloud Native Computing Foundation (CNCF) Landscape

landscape.cncf.io



Stacked Mitigations Instead of Solving the Root Cause?



«Half of the vendors are solving problems Kubernetes helped to create.»

Kubenix (1/2)

- implements K8s OpenAPI specification
  - significantly reduced boiler-plate code
  - type safety during build time

```
{ kubenix, ... }:
let
  name = "nginx";
  version = "1.31.6";
in {
  imports = [ kubenix.modules.k8s ];
  kubernetes.resources.pods.${name}.spec.containers.${name} = {
    image = "${name}:${version}-alpine";
    ports.http.containerPort = 80;
  };
}
```

Kubenix (2/2)

```
{ kubenix, ... }:
let
  app = "nginx";
  port = 80;
  labels = { inherit app; };
in {
  imports = [ kubenix.modules.k8s kubenix.modules.helm ];
  kubernetes.resources = {
    deployments.${app}.spec = { … };
    services.${app}.spec = { … };
    httpRoutes.${app}.spec = { … };
  };
  kubernetes.helm.releases = {
    cilium = {
      chart = kubenix.lib.helm.fetch { … };
      values = { … };
    };
  };
}
```

- leverages NixOS modules to define K8s manifests
  - inheritance and composability
  - interface with Helm ecosystem
    - extend Helm chart functionality
    - collision-aware merge of K8s resources
    - not bound to Helm chart's values

OCI: Docker vs. Nix

```
FROM alpine

RUN apk --no-cache add nginx

ENTRYPOINT ["/usr/sbin/nginx"]

CMD ["-g", "daemon off;"]
```

- imperative steps
- non-reproducible build
- scope of dependencies per image
- tends to not being minimal

```
{ pkgs ? import <nixpkgs> { } }:
pkgs.dockerTools.buildLayeredImage {
  name = "nginx";
  tag = "v${pkgs.nginx.version}";
  config = {
    Entrypoint = [
      "${pkgs.nginx}/bin/nginx"
    ];
    Cmd = [ "-g" "daemon off;" ];
  };
}
```

- declarative and stateless
- reproducible build
- dependencies sharable accross images
- minimal per design

OCI: `buildLayeredImage` Reuses Shared Dependencies

```
pkgs.dockerTools.buildLayeredImage {
  name = "nginx";
  tag = "v${pkgs.nginx.version}";
  config = {
    Entrypoint = [
      "${pkgs.nginx}/bin/nginx"
    ];
    Cmd = [ "-g" "daemon off;" ];
  };
}
```

```
$ podman save --format=oci-dir nginx:v1.30.5
├ …
└ blobs/sha256
   ├ 7127…7418 …/bin/nginx
   ├ …
   ├ 12ce…4609 …/lib/libxml2.so
   ├ 438b…83c0 …/lib/libpcre2-posix.so
   └ 6077…8ba7 …/lib/libc.so
```

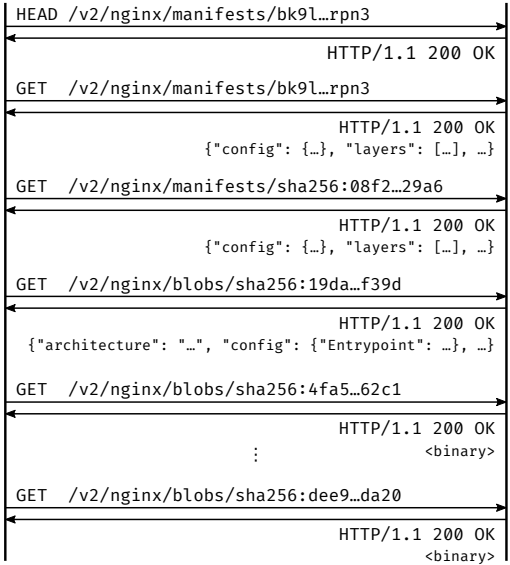
```
pkgs.dockerTools.buildLayeredImage {
  name = "redis";
  tag = "v${pkgs.redis.version}";
  config = {
    Entrypoint = [
      "${pkgs.redis}/bin/redis-server"
    ];
    Cmd = [ "--port" "6379" ];
  };
}
```

```
$ podman save --format=oci-dir redis:v8.10.1
├ …
└ blobs/sha256
   ├ be46…01c5 …/bin/redis-server
   ├ …
   ├ 12ce…4609 …/lib/libxml2.so
   ├ 438b…83c0 …/lib/libpcre2-posix.so
   └ 6077…8ba7 …/lib/libc.so
```

OCI: Integration of `buildLayeredImage`

gitlab.com/arik.grahl/nix-oci

```yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: example.com/
        nginx:bk9l…rpn3
```




```
/nix/store/bk9l…rpn3-nginx.tar.gz
├─ oci-layout
├─ index.json
└─ blobs
   └─ sha256
      ├─ 08f2…29a6
      ├─ 19da…f39d
      ├─ 4fa5…62c1
      ├─ 5253…fa1c
      ├─ 60de…a117
      ├─ 9759…9caa
      ├─ cde9…f780
      ├─ cfad…89f5
      └─ dee9…da20
```

```
pkgs.dockerTools.buildLayeredImage {
  name = "nginx";
  tag = "v${pkgs.nginx.version}";
  config = {
    Entrypoint = [
      "${pkgs.nginx}/bin/nginx"
    ];
    Cmd = [ "-g" "daemon off;" ];
  };
}
```

Monorepository: `flake.nix`

```
{
  inputs = {
    kubenix.url = "github:hall/kubenix";
    cert-manager = {
      url = "https://charts.jetstack.io/charts/cert-manager-v1.20.2.tgz";
      flake = false;
    };
  };
  outputs = {
    packages.${system}.production =
      inputs.kubenix.evalModules.${system} {
        specialArgs = { inherit system inputs; };
        module =
          { kubenix, ... }:
          {
            imports = [
              ./modules/k8s/cert-manager.nix
            ];
            cert-manager.enable = true;
          };
      }.config.kubernetes.resultYAML;
  };
}
```

- upstream dependencies as `inputs`
  - tooling (e.g. Kubenix)
  - Helm charts (e.g. cert-manager)
  - software repositories
- composition layers as `outputs`
  - evaluate NixOS modules
  - flexible and safe configuration
  - produce structured documents

Monorepository: `modules/cert-manager.nix` (1/2)

```nix
{ config, lib, pkgs, system, inputs, ... }:
{
  options.cert-manager = {
    enable = lib.mkEnableOption "cert-manager";
    namespace = lib.mkOption {
      type = lib.types.str;
      default = "cert-manager";
    };
  };
  config = {
    crds = {
      "cert-manager.io".enable = true;
      "gateway.networking.k8s.io".enable = true;
    };
    oci.images = lib.mkMerge [
      (lib.genAttrs [
        "cert-manager-cainjector"
        "cert-manager-controller"
      ] (name: import (./images + "/${name}.nix") ({ inherit pkgs system name; })))
    ]; # …
  };
}
```

- module `options`
  - encapsulate the package's configuration
- module `config`
  - defines dependencies through different modules
    - Custom Resource Definitions (CRDs)
    - OCI images

Monorepository: modules/cert-manager.nix (2/2)

```nix
{ config, lib, pkgs, system, inputs, ... }:
{ # …
  config = { # …
    kubernetes = {
      helm.releases = lib.mkMerge [
        (lib.mkIf config.cert-manager.enable {
          cert-manager = {
            chart = derivation { … };
            values = { … };
          };
        })
      ];
      resources = lib.mkMerge [ (lib.mkIf config.cert-manager.enable {
        deployments = lib.genAttrs [ "cert-manager-cainjector" "cert-manager-controller" ]
          (name: {
            spec.template.spec.containers.${name}.image =
              lib.mkForce config.oci.resolved.${name}.url;
          });
      }) ];
    };
  };
}
```

- module `config`
  - package payload
    - expose an evaluated Helm chart
    - reference defined OCI images

Monorepository: `modules/oci.nix` (1/2)

```nix
{ config, lib, pkgs, system, inputs, ... }:
{
  options.oci = {
    registryBaseURI = lib.mkOption {
      type = lib.types.str;
      default = "registry.example.com";
    };
    images = lib.mkOption {
      type = lib.types.attrsOf lib.types.path;
      default = { };
    };
    resolved = lib.mkOption {
      internal = true;
      type = lib.types.attrsOf (
        lib.types.submodule {
          options = { … }; # archive, name, hash, repsitory, tag, url
        }
      );
      default = { };
    };
  }; # …
}
```

- module `options`
  - configure `registryBaseURI`
  - register OCI images via `images`
    - e.g. `buildLayeredImage`
  - expose registered artifacts as `resolved`

Monorepository: `modules/oci.nix` (2/2)

```nix
{ # …
  config.oci = {
    resolved =
    resolved = lib.mapAttrs (name: archive:
      let
        matches = builtins.match
          "^/nix/store/([^-]+)-(.+)\.tar\.gz$"
          (toString archive);
      in
      {
        inherit archive;
        hash = builtins.elemAt matches 0;
        tag = builtins.elemAt matches 0;
        name = builtins.elemAt matches 1;
        repository = "${cfg.registryBaseURI}/${builtins.elemAt matches 1}";
        url = "${cfg.registryBaseURI}/${builtins.elemAt matches 1}"
              + ":${builtins.elemAt matches 0}";
      }
    ) config.oci.images;
  };
}
```

- module `config`
  - construct `resolved` OCI images
    - consume registered images
    - parses archive's filename structure
    - structurally expose data
    - takes `registryBaseURI` into consideration

Coherent and Simplified Tooling for Kubernetes (1/2)

```
$ nix flake show git.example.com/k8s#production

├ nixosModules
│  ├ cert-manager: NixOS module
│  ├ "crd-cert-manager.io": NixOS module
│  └ "crd-gateway.networking.k8s.io": NixOS module
└ packages
   └ x86_64-linux
      ├ oci: package 'oci'
      ├ oci-cert-manager-cainjector: package '….tar.gz'
      ├ oci-cert-manager-controller: package '….tar.gz'
      └ production: package 'kubenix-generated.yaml'
```

Coherent and Simplified Tooling for Kubernetes (2/2)

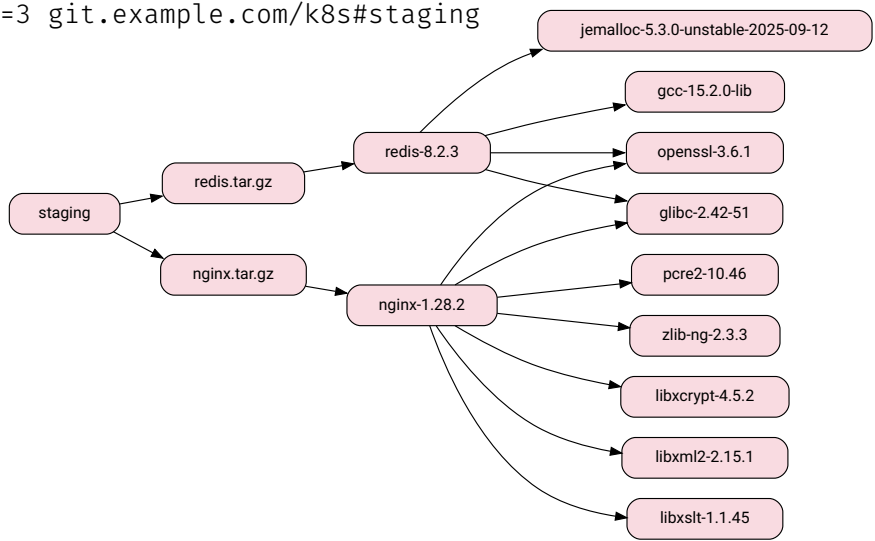
```
$ nix build git.example.com/k8s#production
```

```
$ kubectl apply -f ./result
```

```
customresourcedefinition.apiextensions.k8s.io/certificates.cert-manager.io created
mutatingwebhookconfiguration.admissionregistration.k8s.io/cert-manager-webhook created
deployment.apps/cert-manager created
deployment.apps/cert-manager-cainjector created
configmap/designate-certmanager-webhook-apiservice created
service/cert-manager created
serviceaccount/cert-manager created
clusterrole.rbac.authorization.k8s.io/cert-manager-controller-certificates created
clusterrolebinding.rbac.authorization.k8s.io/cert-manager-controller-certificates created
role.rbac.authorization.k8s.io/cert-manager:leaderelection created
rolebinding.rbac.authorization.k8s.io/cert-manager:leaderelection created
[…]
```

Software Supply Chain: Dependency Graph

```
$ nixgraph --depth=3 git.example.com/k8s#staging
```



Software Supply Chain: SBOM and SLSA

github.com/tiiuae/sbomnix

```
$ sbomnix git.example.com/k8s#staging
```

```yaml
---
bomFormat: CycloneDX
specVersion: '1.4'
version: 1
serialNumber: urn:uuid:30f2…fa39
components:
  - type: library
    bom-ref: /nix/store/rs52…m0×a-nginx-1.28.2.drv
    name: nginx
    version: 1.28.2
    # …
  - type: library
    bom-ref: /nix/store/zf7w…28dp-glibc-2.42-51.drv
    name: glibc
    version: 2.42-51
    # …
  - type: library
    bom-ref: /nix/store/vmyc…hc89-openssl-3.6.1.drv
    name: openssl
    version: 3.6.1
    # …
  - type: library
    bom-ref: /nix/store/109v…5cxy-pcre2-10.46.drv
    name: pcre2
    version: '10.46'
# …
```

```
$ provenance git.example.com/k8s#staging
```

```yaml
---
_type: https://in-toto.io/Statement/v1
predicateType: https://slsa.dev/provenance/v1
predicate:
  buildDefinition:
    resolvedDependencies:
      - name: nginx-1.28.2
        uri: /nix/store/4lb0…fxlv-nginx-1.28.2
        digest:
          sha256: 11be…0392
        annotations:
          version: 1.28.2
      - name: glibc-2.42
        uri: /nix/store/1xl0…d50m-glibc-2.42.drv
        digest:
          sha256: b6cb…92d3
        annotations:
          version: '2.42'
      # …
      - name: gcc-15.2.0
        uri: /nix/store/6vxm…bfdv-gcc-15.2.0.drv
        digest:
          sha256: ad38…6a52
        annotations:
          version: 15.2.0
# …
```

Conclusion (1/3)



- Kubenix
  - type-safe definition of Kubernetes resources
  - reduces boiler-plate code
  - interfaces with Helm

Conclusion (2/3)



- Nix-based OCI containers
  - declarative and reproducible build
  - share dependencies accross OCI images
  - potentially small OCI images

Conclusion (3/3)



- NixOS module system
  - streamlines composability
  - end-to-end definition
  - strong software supply chain security (SSCS)

Questions & Answers



arik-grahl.de/talks/nixcon-2026



arik-grahl.de/tags/nix