arik-grahl.de/talks/containerdays-2025

Kubenix

Declare Your K8s Workloads Fully Reproducible

echo $(whoami)

```yaml
---
apiVersion: v1
kind: Person
metadata:
  name: Arik Grahl
  pronouns: he/him
spec:
  work:
    company: SysEleven
    role: Senior Software Engineer
  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
```

YAML

```yaml
---
a: &anchor
  x: baz
  y: 1.23
b: *anchor
c:
  <<: *anchor
  y: 4.56
d: { "x": "foo", "y": 7.89 }
e:
  - [true, NO]
  - [1, -1, 1.2, 1e6]
  - [1.2.3, foo, "bar"]
```

- verbose and repetitive
- not easily extensible
- challenging inheritance
- type confusion

Tooling to Generate Structured Documents

- tooling to generate structured documents
  - Jsonnet
  - Kustomize
  - Helm and Helmfile
- instead of solving the root cause
  - just mitigating the problem
  - another level of indirection

Helm (1/2)

- defacto standard for packaging
- rudimentary dependency management for downstream charts
- heavily based upon Golang's `text/template`

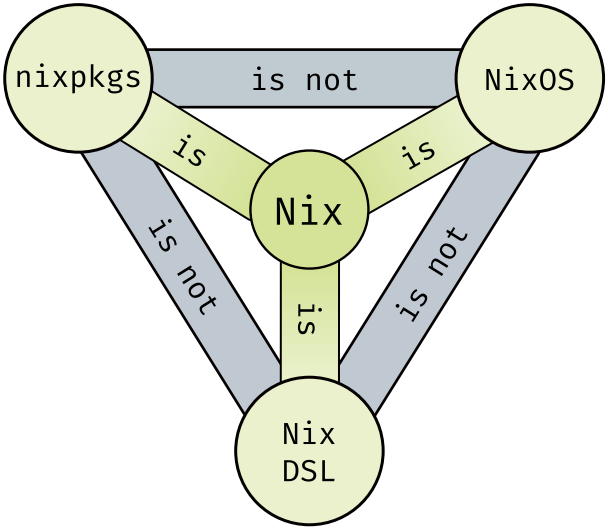
Helm (2/2)

```
{{- if .Values.rbac.create }}
{{- if and .Values.rbac.scope (not .Values.controller.scope.enabled) -}}
  {{ required "Invalid configuration …" (index (dict) ".") }}
{{- end }}
{{- if not .Values.rbac.scope -}}
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    {{- include "ingress-nginx.labels" . | nindent 4 }}
    {{- with .Values.controller.labels }}
    {{- toYaml . | nindent 4 }}
    {{- end }}
  name: {{ include "ingress-nginx.fullname" . }}
# …
{{- end }}
{{- end }}
```

Entering Nix

- Nix enables systems, which are
  - reproducible
  - declarative
  - reliable

What is Nix?



- NixOS
  - Linux-based OS on top of Nixpkgs
  - immutable design
  - atomic update model
- Nixpkgs
  - package manager
  - large mono repository
  - collection of over 100k packages
- Nix DSL
  - purely functional
  - lazily evaluated
  - purpose-built

Nix DSL: Purely Functional

- any valid piece of Nix code is an expression returning a value

```
"foo"           # ⇒ "foo"
{ x = "foo"; } # ⇒ { x = "foo"; }
```

- evaluating a Nix expression yields a data structure

```
let f = x: x; in f "foo"           # ⇒ "foo"
let f = x: x; in f { x = "foo"; } # ⇒ { x = "foo"; }
```

- evaluation does not execute a sequence of operations

```
let x = 1; y = x + 1; in y # ⇒ 2
let y = x + 1; x = 1; in y # ⇒ 2
```

Nix DSL: Lazily Evaluated

- expressions are only evaluated if result is requested

```
let a = {
  x = builtins.throw "exception";
  y = "foo";
}; in a.x                          # ⇒ error: exception
```

```
let a = {
  x = builtins.throw "exception";
  y = "foo";
}; in a.y                          # ⇒ "foo"
```

Nix DSL: Purpose-Built

- domain specific language (DSL) for the Nix package manager

```
let pkgs = import <nixpkgs> {}; in
pkgs.buildGoModule rec {
  pname = "kubectl";
  version = "1.33.4";
  subPackages = [ "cmd/${pname}" ];
  vendorHash = null;
  src = pkgs.fetchFromGitHub {
    owner = "kubernetes";
    repo = "kubernetes";
    rev = "v${version}";
    hash = "sha256-KENE…8vlQ=";
  };
}
```

Introducing Kubenix

- Kubenix: definition of K8s manifests via NixOS modules
  - significantly reduced boiler-plate code
  - type safety
  - inheritance and composability
  - interface with Helm and OCI ecosystems

Defining a K8s Manifest with Plain Nix (1/2)

```nix
{ pkgs ? import <nixpkgs> { } }:
let
  name = "nginx";
  version = "1.29.1";
in
pkgs.writeText "pod.yaml" (pkgs.lib.generators.toYAML { } {
  apiVersion = "v1";
  kind = "Pod";
  metadata = { inherit name; };
  spec.containers = [{
    inherit name;
    image = "${name}:${version}-alpine";
    ports = [{
      name = "http";
      containerPort = 80;
    }];
  }];
})
```

Defining a K8s Manifest with Plain Nix (2/2)

```nix
{ pkgs ? import <nixpkgs> { } }:
let
  name = "nginx";
  version = "1.29.1";
in
pkgs.writeText "pod.yaml" (pkgs.lib.generators.toYAML { } {
  apiVersion = "v1";
  kind = "Pod";
  metadata = { inherit name; };
  spec.containers = [{
    inherit name;
    image = "${name}:${version}-alpine";
    ports = [{
      name = "http";
      containerPort = "EIGHTY"; # ⇒ cannot unmarshal string into type int32
    }];
  }];
})
```

The OpenAPI Specification: A Standard to Describe HTTP APIs

```yaml
definitions:
  # …
  io.k8s.api.core.v1.Pod:
    description: "Pod is a collection of containers that can run on a host. This […]"
    properties:
      apiVersion:
        description: "APIVersion defines the versioned schema of this representation […]"
        type: string
      kind:
        description: "Kind is a string value representing the REST resource this […]"
        type: string
    # …
  io.k8s.api.core.v1.ContainerPort:
    description: "ContainerPort represents a network port in a single container."
    properties:
      containerPort:
        default: 0
        description: "Number of port to expose on the pod's IP address. This must […]"
        format: int32
        type: integer
    # …
```

Kubenix Implements the K8s OpenAPI Specification (1/2)

- significantly reduced boiler-plate code

```
{ kubenix, ... }:
let
  name = "nginx";
  version = "1.29.1";
in {
  imports = [ kubenix.modules.k8s ];
  kubernetes.resources.pods.${name}.spec.containers.${name} = {
    image = "${name}:${version}-alpine";
    ports.http.containerPort = 80;
  };
}
```

Kubenix Implements the K8s OpenAPI Specification (2/2)

- type safety

```
{ kubenix, ... }:
let
  name = "nginx";
  version = "1.29.1";
in {
  imports = [ kubenix.modules.k8s ];
  kubernetes.resources.pods.${name}.spec.containers.${name} = {
    image = "${name}:${version}-alpine";
    ports.http.containerPort = "EIGHTY";
  };
} # ⇒ error: containerPort is not of type signed integer
```

Defining a Webapplication with Kubenix (1/4)

```
{ kubenix, ... }:
let
  app = "nginx";
  version = "1.29.1";
  port = 80;
  labels = { inherit app; };
in {
  imports = [ kubenix.modules.k8s ];
  kubernetes.resources = {
    deployments.${app}.spec = { … };
    services.${app}.spec = { … };
    httpRoutes.${app}.spec = { … };
  };
}
```

Defining a Webapplication with Kubenix (2/4)

```
{ kubenix, ... }:
let
  app = "nginx";
  version = "1.29.1";
  port = 80;
  labels = { inherit app; };
in { # …
  deployments.${app}.spec = {
    selector.matchLabels = labels;
      template = {
        metadata = { inherit labels; };
        spec.containers.${app}.image = "${app}:${version}-alpine";
      };
  }; # …
}
```

Defining a Webapplication with Kubenix (3/4)

```
{ kubenix, ... }:
let
  app = "nginx";
  version = "1.29.1";
  port = 80;
  labels = { inherit app; };
in { # …
  services.${app}.spec = {
    selector = labels;
    ports = [{
      inherit port;
      protocol = "TCP";
    }];
  }; # …
}
```

Defining a Webapplication with Kubenix (4/4)

```
{ kubenix, ... }:
let
  app = "nginx";
  version = "1.29.1";
  labels = { inherit app; };
  port = 80;
in { # …
  httpRoutes.${app}.spec = {
    hostnames = [ "example.com" ];
    rules = [{
      matches = [{
        path = { type = "PathPrefix"; value = "/"; };
      }];
      backendRefs = [{
        inherit port;
        name = app;
      }];
    }];
  }; # …
}
```

Consuming a Helm Chart with Kubenix (1/2)

```
{ kubenix, ... }:
{
  imports = [ kubenix.modules.helm ];
  kubernetes.helm.releases.cilium = {
    chart = kubenix.lib.helm.fetch {
      repo = "https://helm.cilium.io";
      chart = "cilium";
      version = "1.18.1";
      sha256 = "sha256-baUV…G71U=";
    };
    values = {
      image.pullPolicy = "IfNotPresent";
      ipam.mode = "kubernetes";
      operator.replicas = 1;
      kubeProxyReplacement = true;
      gatewayAPI = {
        enabled = true;
        hostNetwork.enabled = true;
      };
    };
  };
}
```

Consuming a Helm Chart with Kubenix (2/2)

```
{ kubenix, ... }:
let
  app = "nginx";
  version = "1.29.1";
  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 = { … };
    };
  };
}
```

- collision-aware merge of K8s resources
- extend Helm chart functionality
- dependency management for K8s
- not bound to Helm's templating and values

Building and Applying

```
$ nix build -f k8s.nix -o result
```

```
$ jq -c < ./result '.items[] | [.apiVersion, .kind]' | uniq -c

2 ["apps/v1","Deployment"]
3 ["v1","Service"]
1 ["gateway.networking.k8s.io/v1","HTTPRoute"]
…
```

```
$ kubectl apply -f ./result

deployment.apps/cilium-operator created
deployment.apps/nginx created
service/cilium-envoy created
service/hubble-peer created
service/nginx created
httproute.gateway.networking.k8s.io/nginx created
…
```

- one tool to build everything
- reproducible and pure builds
- GitOps friendly
- flexible towards custom tooling

End-To-End Definition of a K8s Workloads: Container

```
{ kubenix, ... }:
let
  name = "nginx";
  version = pkgs.nginx.version;
  container = pkgs.dockerTools.buildLayeredImage {
    inherit name;
    tag = "v${version}";
    config.Entrypoint = [ "${pkgs.nginx}/bin/nginx" ];
    config.Cmd = [ "-g" "daemon off;" ];
  };
in {
  imports = [ kubenix.modules.k8s ];
  kubernetes.resources.pods.${name}.spec.containers.${name} = {
    image = "registry.internal/${container}";
  }
};
```

End-To-End Definition of a K8s Workloads: SBOM and SLSA

```
$ nix profile install github:tiiuae/sbomnix
```

```
$ sbomnix --depth=2 ./result
```

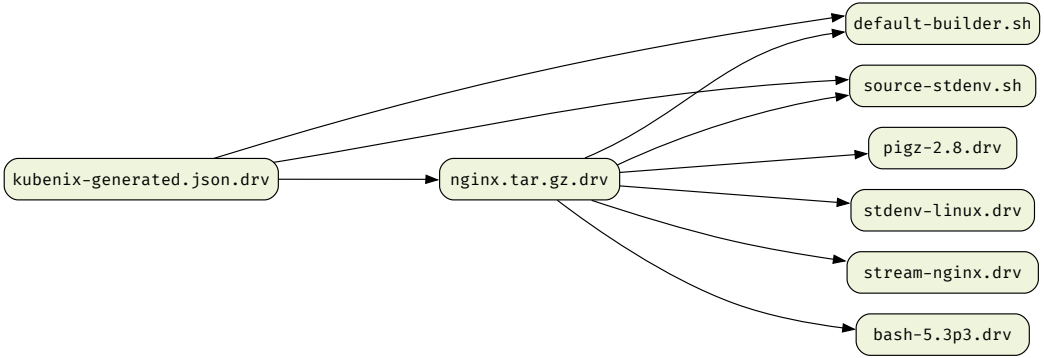
```
spdxVersion: SPDX-2.3
dataLicense: CC0-1.0
SPDXID: SPDXRef-DOCUMENT
name: SPDXRef-nix-store-…
documentNamespace: sbomnix://…
packages:
  - name: glibc
    SPDXID: SPDXRef-nix-store-…
    versionInfo: 2.40-66
    externalRefs:
      - referenceCategory: SECURITY
        referenceType: cpe23Type
        referenceLocator: cpe:2.3:a:glibc:…
      - referenceCategory: PACKAGE-MANAGER
        referenceType: purl
        referenceLocator: pkg:nix/glibc@2.40-66
  - name: nginx
    SPDXID: SPDXRef-nix-store-…
    versionInfo: 1.28.0
    externalRefs:
      - referenceCategory: SECURITY
        referenceType: cpe23Type
  # …
```

```
$ provenance --recursive ./result
```

```
_type: https://in-toto.io/Statement/v1
subject:
  - name: out
    uri: /nix/store/…
    digest:
      sha256: 0v60…m32y
predicateType: https://slsa.dev/provenance/v1
predicate:
  buildDefinition:
    resolvedDependencies:
      - name: glibc-2.40-66-getent
        uri: /nix/store/…
        digest:
          sha256: 0hbr…xfcj
      - name: nginx-1.28.0
        uri: /nix/store/…
        digest:
          sha256: 1hd2…2l0s
      - name: nginx-1.28.0.tar.gz
        uri: /nix/store/…
        digest:
          sha256: 09sq…n4b4
      # …
```

End-To-End Definition of a K8s Workloads: Dependency Graph

```
$ nixgraph --depth=2 ./result
```



Conclusion

- Nix
  - purely functional, lazily evaluated, and purpose-built language
  - drives Nixpkgs as a large and attractive ecosystem

- Kubenix
  - reduces boilerplate code and provides type safety via OpenAPI spec
  - enables inheritance and composability of complex stacks
  - makes use of a large Helm and OCI ecosystem

Package management for the *platform to build platforms*.

Questions & Answers

arik-grahl.de/talks/containerdays-2025

github.com/hall/kubenix

sys11.it/mka