



# KUBERNETES AVEC DHALL

TRISTAN GOSSELIN-HANE

# DHALL

C'EST QUOI?



**Langage de programmation fonctionnel orienté pour la configuration**

**Peut être directement consommé par des applications ou être exporté vers plusieurs autres formats comme JSON, YAML, TOML etc.**

**Fortement typé**

**Axé sur les principes DRY et le refactoring, excellents messages d'erreur**

**Inspiré du langage de configuration Nix, implémenté en Haskell**

**Validation intégrée**

# **CARACTÉRISTIQUES**

**Records, Types, Unions, Fonctions**

**Librairie standard**

**Currying/Application partielle**

**Imports (“package management”)**

# DHALL + KUBERNETES

COMMENT EST-CE QUE LES DEUX SE REJOIGNENT?



**Génération de Types Dhall pour Kubernetes à partir de sa spécification OpenAPI**

**Configuration validée par le type system**

**Sortie de la configuration en YAML, prête à être consommée par l'API de Kubernetes ou bien d'autres outils qui opèrent sur du YAML.**

**<https://github.com/dhall-lang/dhall-kubernetes>**

# DHALL VS ALTERNATIVES

DANS LE CONTEXTE DE GÉNÉRATION DE MANIFESTS

**Jsonnet:** Similaire, manque de types

**Kustomize:** Très verbose, manque de substitution de variables

**Helm:** Templating textuel d'un langage sensible au whitespace, très verbose et ouvert aux erreurs

# HELM?



Templating  
YAML

Generating  
JSON



# LIMITATIONS/IRRITANTS RENCONTRÉES

## Impossible d'étendre des unions existantes

```
let KubernetesManifest =
  -- Our own union for kubernetes manifests because the union
  -- from dhall-kubernetes cannot be extended to add CRDs
  -- (SecretProviderClass in this case)
  < Deployment : kubernetes.Deployment.Type
  | SecretProviderClass : SecretProviderClass.Type
  | ConfigMap : kubernetes.ConfigMap.Type
  | Service : kubernetes.Service.Type
  | Ingress : kubernetes.Ingress.Type
  >
```

## Pas de list comprehension

```
let makeAzureKeyVaultSecretProviderClasses
  : DotNetApplication.Type -> List KubernetesManifest
  = \(app : DotNetApplication.Type) ->
    Prelude.List.map
      KeyVaultSecrets.Type
      KubernetesManifest
      makeAzureKeyVaultSecretProviderClass
      app.keyVaultSecrets
```



# SURVOL

## LIBRAIRIE

```
let DotNetApplication =
  { Type =
    { name : Text
    , image : Image.Type
    , replicas : Natural
    , nodePool : Text
    , config : List ConfigElement.Type
    , keyVaultSecrets : List KeyVaultSecrets.Type
    , webConfiguration : WebConfiguration.Type
    }
  , default =
    { replicas = 1
    , nodePool = "default"
    , webConfiguration = WebConfiguration::={}
    }
  }
```

TYPE

```
let makeDotNetApplication
  : DotNetApplication.Type -> List KubernetesManifest
= \app : DotNetApplication.Type ->
  [ KubernetesManifest.Deployment (deployment app)
  , KubernetesManifest.ConfigMap (makeConfigMap app)
  , KubernetesManifest.Service (makeWebService app)
  ]
  # ( if app.webConfiguration.enabled
  then [ KubernetesManifest.Ingress (makeWebIngress app) ]
  else [] : List KubernetesManifest
  )
  # makeAzureKeyVaultSecretProviderClasses app

in { makeDotNetApplication
, DotNetApplication
, ConfigElement
, KeyVaultSecrets
, Image
, WebConfiguration
}
```

FONCTION

"EXPORTS"

DEFINITION

## APPLICATION

```
let sample =
  lib.DotNetApplication::{
    , name = "sample-function-app"
    , image = lib.Image::{
      , registry = "acrhiloshared01.azurecr.io"
      , name = "sys-aks-poc/webjobs"
      , tag = "exampleTag"
      }
    , replicas = 3
    , nodePool = "systemsspot"
    , webConfiguration = lib.WebConfiguration::{
      , enabled = True
      , tls = True
      , hosts = [ "demo.dev.hilo.private" ]
      }
    }

in lib.makeDotNetApplication sample
```

APPLICATION DE LA FONCTION

# DEMOTIME

