

設定記述言語 CUE で YAML Hell に立ち向かえ

チェシャ猫 (@y_taka_23)

Kubernetes Meetup Tokyo #29 (2020/03/26)

Manifest をどう管理するか？

Manifest の差分管理の難しさ

- Kubernetes のアーキテクチャ
 - Fine-grained な分散システム
 - リソース数が多くなり、しかも一貫性が必要
 - 少しだけ違うファイルが多数必要
- 組織に応じた責務の分担
 - K8s に詳しい基盤側とそうでもないアプリ側
 - どこまで任せるか・いかにして統制をとるか



<https://cuelang.org>

```
apiVersion: "apps/v1"
kind: "Deployment"
metadata: name: "myapp"
spec: {
  replicas: 3
  selector: matchLabels: app: "myapp"
  template: {
    metadata: labels: app: "myapp"
    spec: containers: [{
      name: "myapp"
      image: "repo/myapp:v1.0"
      ports: [{
        containerPort: 8080
      }]
    }]
  }
}
```

子要素が一つのみのパスは圧縮可能

CUE によるテンプレートティング

```
deployment: <Name>: {  
  apiVersion: string  
  kind: "Deployment"  
  metadata: name: Name  
  spec: {  
    replicas: *1 | int  
    selector matchLabels app: Name  
    template: {  
      metadata: labels: app: Name  
      spec: containers: [{  
        name: Name  
      }]  
    }  
  }  
}
```

The diagram illustrates the use of parameters and default values in a Kubernetes Deployment manifest. A box labeled "パラメータの利用" (Parameter Usage) has an arrow pointing to the `<Name>` placeholder in the `deployment:` key. Another box labeled "デフォルト値" (Default Value) has an arrow pointing to the `*1` in the `replicas` field, indicating that `1` is the default value for the `int` type.

```
deployment: "myapp": {  
  apiVersion: "apps/v1"  
  spec: {  
    replicas: 3  
    template: spec: containers: [{  
      image: "repo/myapp:v1.0"  
      ports: [{  
        containerPort: 8080  
      }]  
    }]  
  }  
}
```

パラメータへの代入

属性の追加

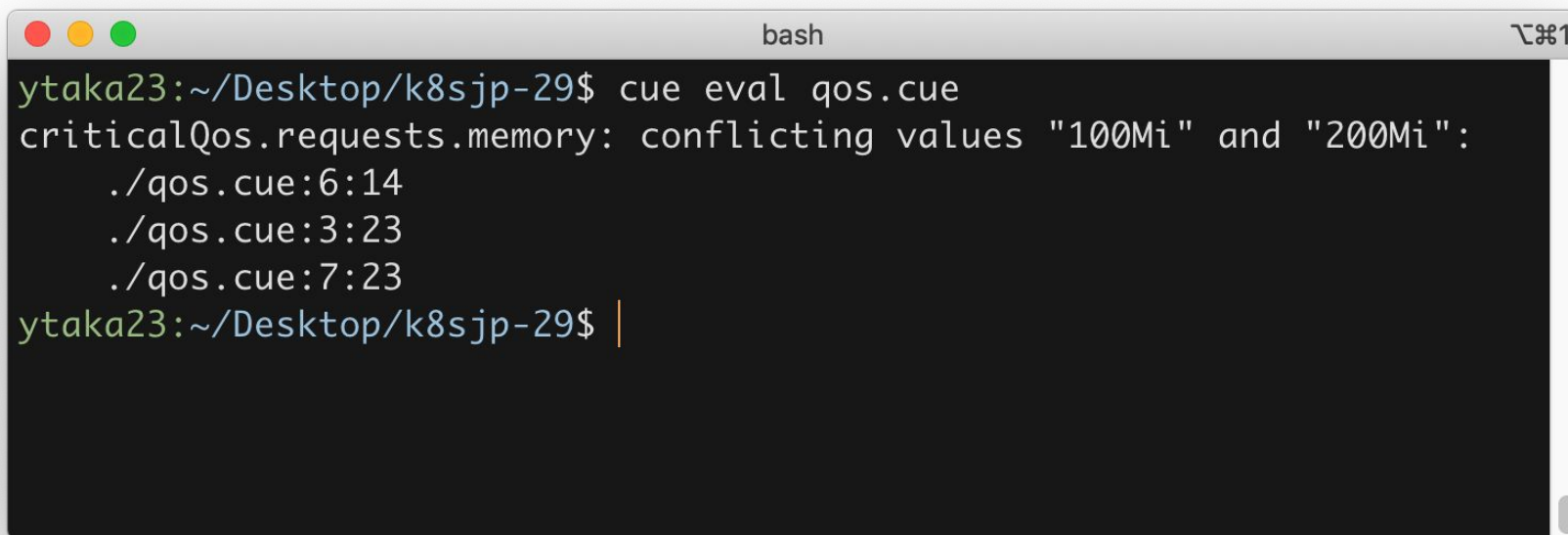
CUE による差分管理

```
defaultQoS: {  
  limits: memory: "200Mi"  
  requests: memory: "100Mi"  
}
```

```
criticalQoS: defaultQoS & {  
  requests: memory: "200Mi"  
}
```

継承先での値の上書き(?)



A terminal window with a title bar containing three colored circles (red, yellow, green) and the text 'bash'. The terminal output shows a user 'ytaka23' at a prompt '~ / Desktop / k8sjp - 29 \$' running the command 'cue eval qos.cue'. The output is a multi-line error message: 'criticalQos.requests.memory: conflicting values "100Mi" and "200Mi":', followed by three indented lines: './qos.cue:6:14', './qos.cue:3:23', and './qos.cue:7:23'. The prompt returns to '~ / Desktop / k8sjp - 29 \$' with a vertical cursor bar.

```
ytaka23:~/Desktop/k8sjp-29$ cue eval qos.cue
criticalQos.requests.memory: conflicting values "100Mi" and "200Mi":
  ./qos.cue:6:14
  ./qos.cue:3:23
  ./qos.cue:7:23
ytaka23:~/Desktop/k8sjp-29$ |
```

一度確定した値は上書きできない！

型かリテラルしか書けないなら
JSON Schema と同程度では？

```
municipality: {  
  name:      string  
  pop:       int  
  capital:   bool  
}
```

Data Schema

```
moscow: {  
  name:      "Moscow"  
  pop:       11.92M  
  capital:   true  
}
```

Concrete Data

```
municipality: {  
  name:      string  
  pop:       int  
  capital:   bool  
}
```

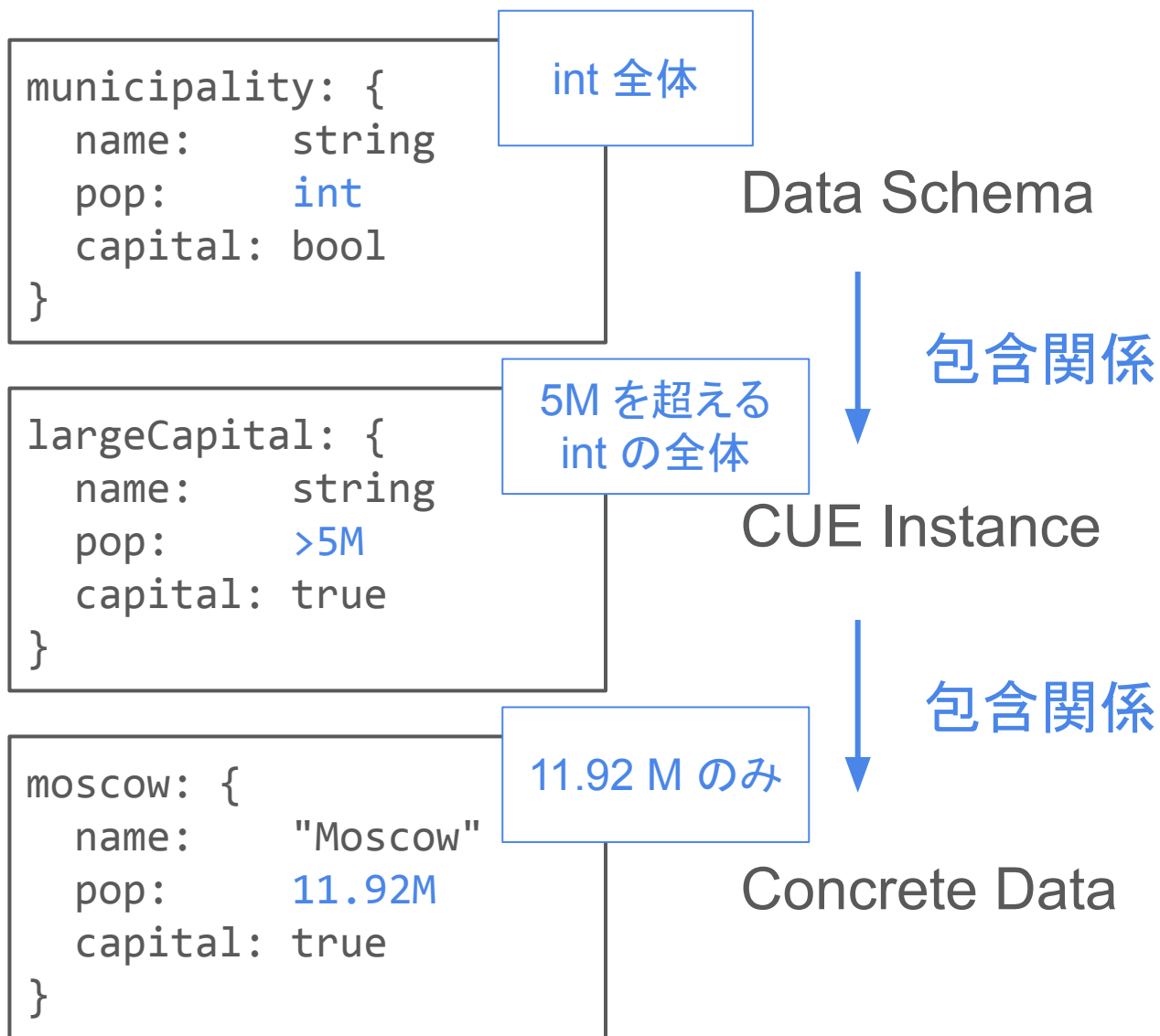
Data Schema

```
largeCapital: {  
  name:      string  
  pop:       >5M  
  capital:   true  
}
```

CUE Instance

```
moscow: {  
  name:      "Moscow"  
  pop:       11.92M  
  capital:   true  
}
```

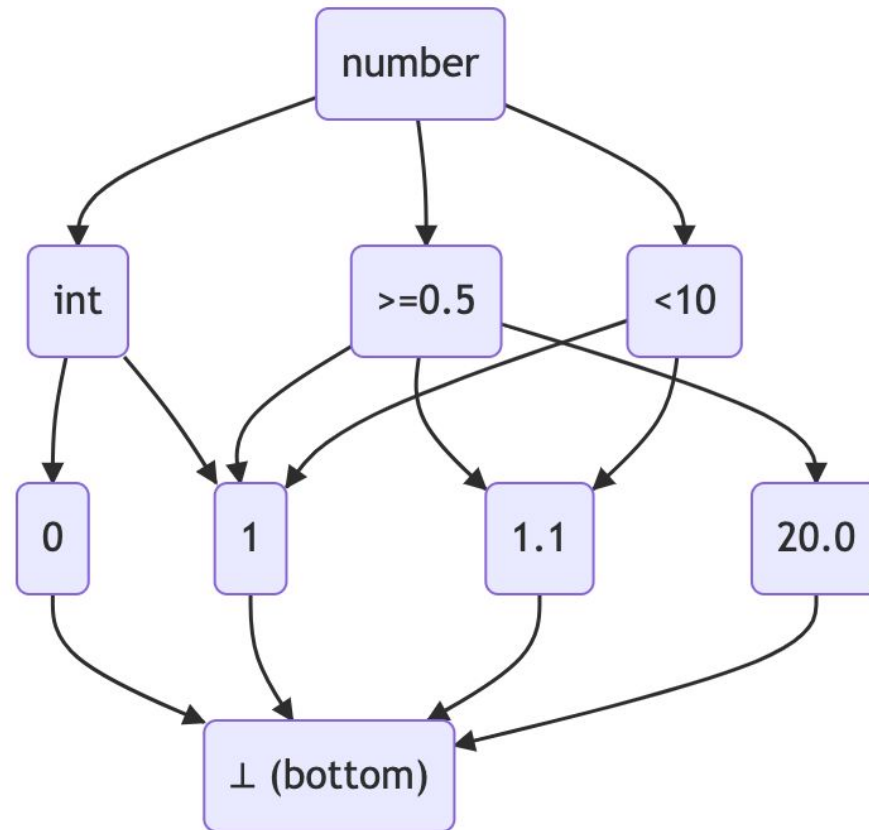
Concrete Data



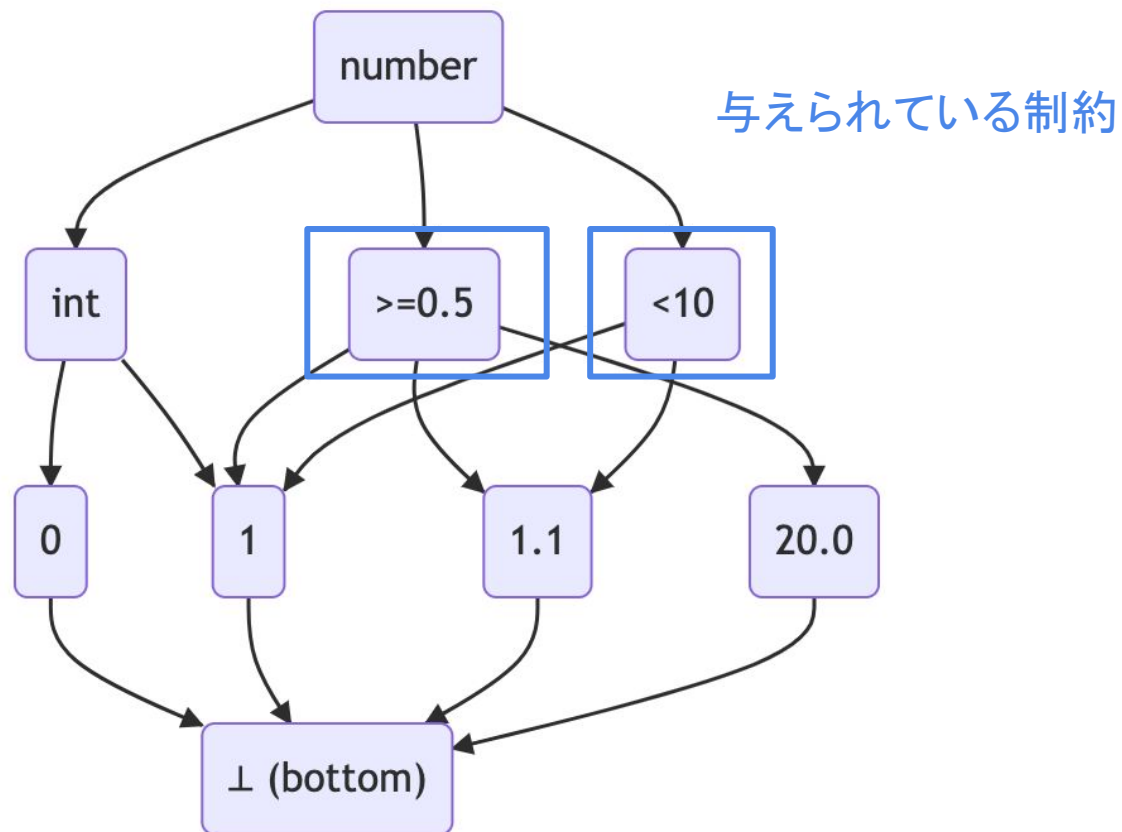
「Types are Values」

類似ツールと CUE の戦略の違い

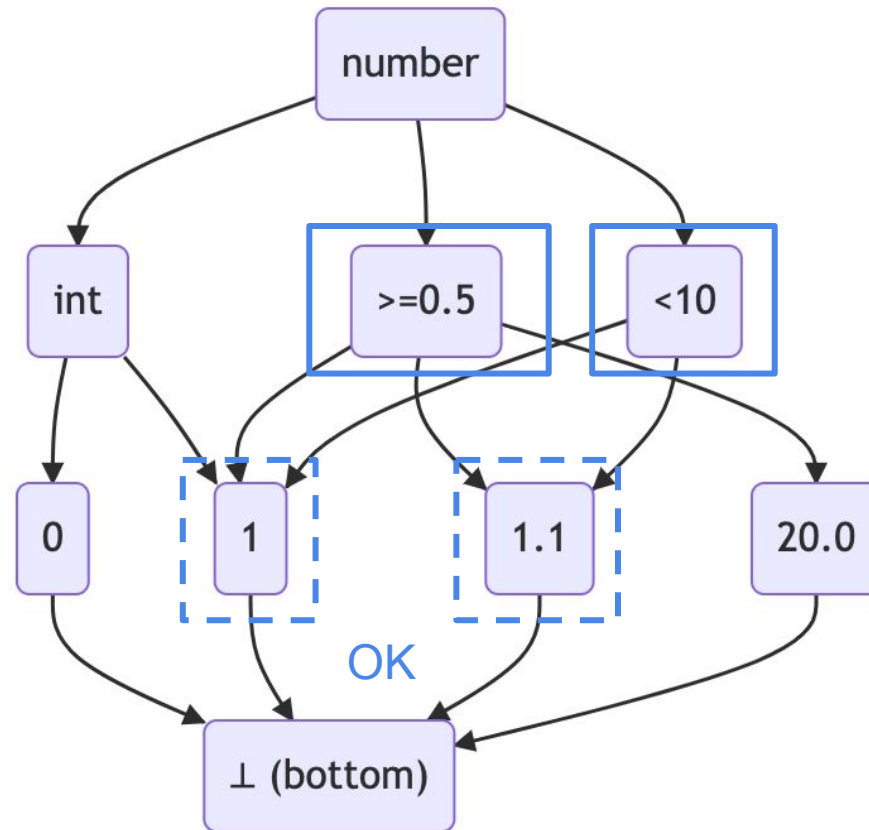
- Kustomize や Jsonnet : 継承ベース
 - ベースの値を上書きすることでカスタマイズ
 - 最終的に有効な値がわかりづらい
- CUE : 制約ベース
 - 複数の条件を重ねがけしてゆくことで合成
 - 確定値は「ちょうどその値になる」という制約
 - 結果は合成順序によらず「一番厳しいもの」



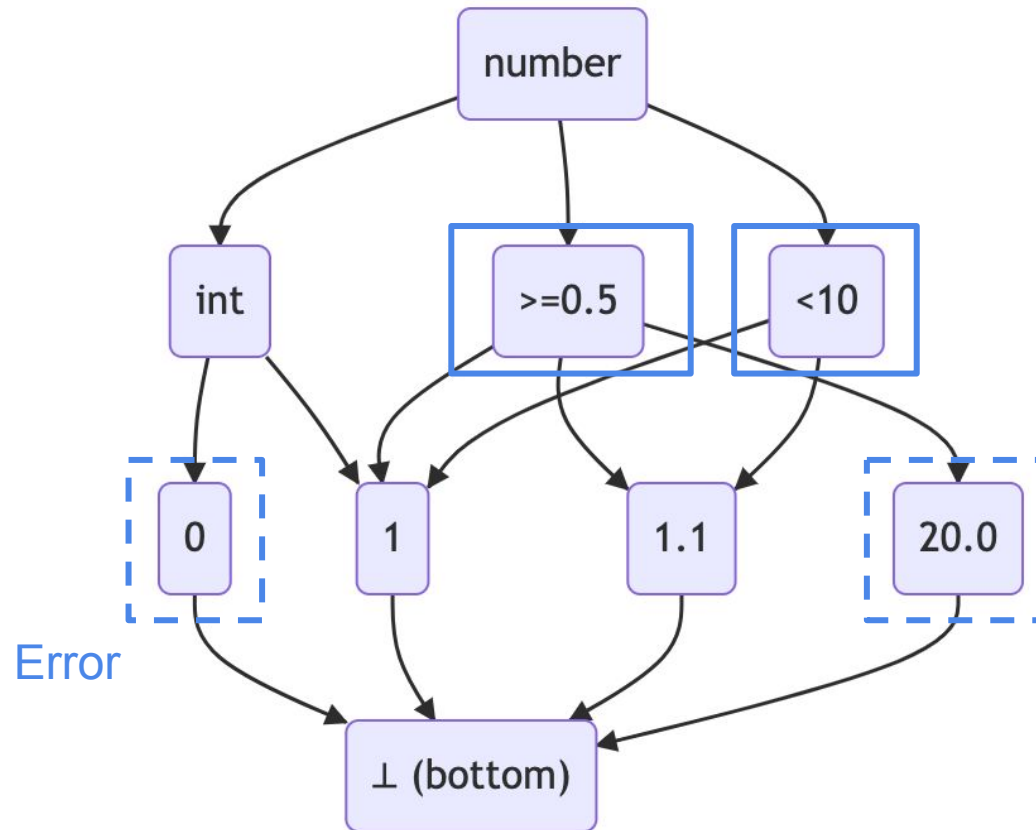
<https://cuelang.org/docs/concepts/logic>



<https://cuelang.org/docs/concepts/logic>



<https://cuelang.org/docs/concepts/logic>



<https://cuelang.org/docs/concepts/logic>

CUE の自動化関連機能

- import / export / trim コマンド
 - 既存の YAML を読み込み、不要部分を削除
- Go との滑らかな連携
 - 構造体から CUE の定義読み込みや SDK
- CUE 内蔵のカスタムコマンド機構
 - 雑多なツール類の増殖を防ぐ

まとめ

- Kubernetes Manifest 管理の難しさ
 - カスタマイズ性と制約の統制のバランス
- CUE による設定ファイルの管理
 - 制約の強さによる包含関係を管理
- 自動化と相性が良い仕組み
 - ツーリングや既存の資産を利用した導入の重視

Yet Another Manifest Life!

Presented by チェシヤ猫 (@y_taka_23)