Cue 0.7

Mise à jour de Cue 0.7

sur env-ng et trackbone

Sommaire

1. Évolutions constatées
2. Problème de dépendances des taches flow
3. Problème de concurrence avec cue flow
4. Rappel sur la fermeture (close) des définitions
5. Problème du modulo

Pourquoi ?

- Cue 0.4 n’est pas thread safe -> bloque nos travaux de parallélisation des taches par zone
- Upgrade en Go 1.20

Évolutions constatées

- Refonte du calcul des dépendances sur les taches flow
- Durcissement sur les struct close, il semble que le moteur avait des “tolérances”, aujourd’hui il ne laisse rien passer
- Changement de l’affichage des float : précision à 1 chiffre après la virgule automatique

Problème de dépendances des taches flow

Rappel sur les tasks flow

- des “pre” et “post” task -> moteur “Cue Flow”
- `$id` et `$after` -> ❌ ~~moteur “Cue Flow”~~

 Cue Flow n’est pas “Cue Tool.Task”

Avec trackbone, on charge ce qui se trouve sous les branches

- `run.pre_tasks`
- `run.pre_apply_tasks`
- `run.pre_plan_tasks`
- `run.pre_destroy_tasks`
- `run.post_tasks`
- `run.post_apply_tasks`
- `run.post_plan_tasks`
- `run.post_destroy_tasks`

Problème dépendance `_cst_id`

Utilisé pour injecter des dépendances auto-magiquement pour les jobs d’un certain type.

C’est défini dans `custom_tasks.cue`

```
#RDSDBSecret: {
        vault.#Read

        // Custom Task ID
        _cst_id: "RDSDBSecret"

        // Interface
        namespace: string

        // The values of the DB secret
        data: {
                // Applications should use the private IP
                private_ip: string
                port:       string
                // The DB name
```

custom_tasks.cue

```
run: [=~"_tasks$"]: [string]: {
    _cst_id: *"" | string
    if _cst_id == #RDSDBSecret._cst_id {
        namespace: string
        url:       run.pre_tasks["vault_\(infra_zone.name)"].
        path:      "secret/zones/\(zone.provider.type)/\(zone

    }
}
```

Problème

`if _cst_id == #RDSDBSecret._cst_id` semble poser problème au calcul des dépendances de Cue Flow et provoque des dépendances illogiques.

Solution

Pour retirer ce `if` on utilise un suffixe sur le job

```
run: [=~"_tasks$"]: [=~"_db$"]: #RDSDBSecret & {
    namespace: string
    url:       run.pre_tasks["vault_\(infra_zone.name)"].url
    path:      "secret/zones/\(zone.provider.type)/\(zone.nam
}
```

Problème dépendance sur interpolation

Lors d’une interpolation, il semble ne pas toujours voir la dépendance.

```
login: exec.#Run & {
    cmd: ["nix-shell", "--quiet", "--run", """
        az login -u \(secret.data.username) -p \(secret.data.
        """,
    ]
}
```

je n’ai pas toujours reproduit ce bug

Solution

On extrait la dépendance pour la rendre explicite

```
login: exec.#Run & {
    let UserName = tasks.secret.data.username
    let Password = tasks.secret.data.password
    cmd: ["nix-shell", "--quiet", "--run", """
        az login -u \(UserName) -p \(Password) --allow-no-sub
        """,
    ]
}
```

Problème dépendance sur une valeur concrète

Si la dépendance est liée à un élément concret, il ne sera pas vu comme dépendance de la tache.

```
url: pre_tasks["vault_infra"].url
```

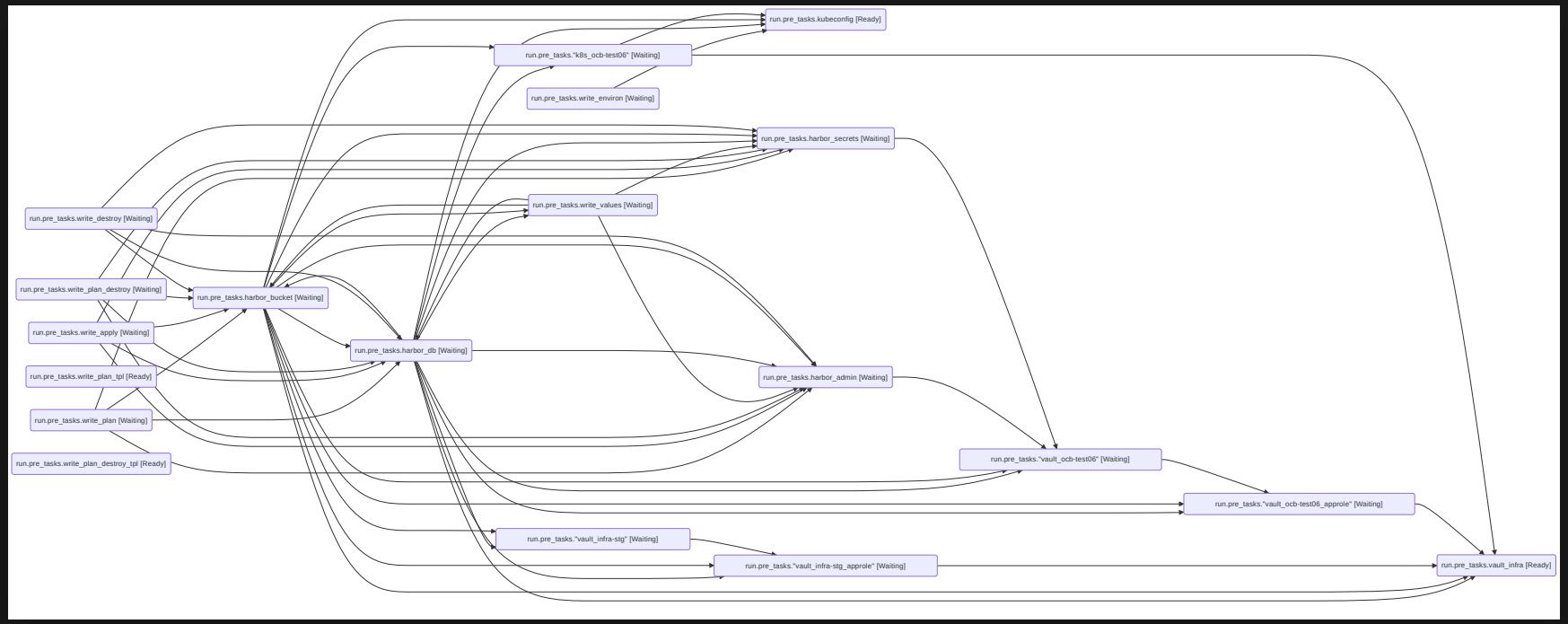
`.url` est défini et ne dépend pas de l'exécution de la tache `vault_infra`.
donc notre tache ne dépend pas de `vault_infra`.

Solution

Pour résoudre ce problème on va lier explicitement la tache `vault_infra` à notre tache.

```
_vault_login: pre_tasks["vault_infra"]
url:          _vault_login.url
```

Exemple Boucle de dépendance



Conséquences / Évolutions

Évolution - Vault

```
url: pre_tasks["vault_\(zone.name)"].url
```

Devient

```
_vault_login: pre_tasks["vault_\(zone.name)"]
url:          _vault_login.url
```

script

```
grep "vault.#Write" -rl | xargs \
sed -ri '/vault\.#Write/,+2 s/^(\s+)url:\s+(.*)\.url/\1_vault

grep "vault.#Read" -rl | xargs \
sed -ri '/vault\.#Read/,+2 s/^(\s+)url:\s+(.*)\.url/\1_vault_

grep "vault.#List" -rl | xargs \
sed -ri '/vault\.#List/,+2 s/^(\s+)url:\s+(.*)\.url/\1_vault_

grep "vault.#Delete" -rl | xargs \
sed -ri '/vault\.#Delete/,+2 s/^(\s+)url:\s+(.*)\.url/\1_vaul
```

Évolution - Kube Config

- Ajout du suffixe `_kubecfg` -> #KubectlAction

```
[=~"_kubecfg$"]: {
    $after: [for task_key, _ in run.pre_tasks if task_key =~
    context: *zone.name | string
    env: KUBECONFIG: run.pre_tasks.kubeconfig.abspath
}
```

Exemple

```
run: pre_tasks: patchNS: #KubectlAction

// devient

run: pre_tasks: patchNS_kubecfg: #KubectlAction
```

Évolution - DB

- Ajout du suffixe _db pour #RDSDBSecret

```
[=~"_db$"]: #RDSDBSecret & {
    _vault_login: run.pre_tasks["vault_\(infra_zone.name)"]
    url:          _vault_login.url
    namespace:    string
    path:         "secret/zones/\(zone.provider.type)/\(zone.
}
```

Exemple

```
let DB = run.pre_tasks.db.data

// devient

let DB = run.pre_tasks.rds_db.data
```

Autre Exemple

```
admin_db_read: #RDSDBSecret & {namespace: "customers-manager'

// devient

admin_read_db: #RDSDBSecret & {namespace: "customers-manager'
```

Évolution - Bucket

- Ajout du suffixe `_bucket` pour `#S3Secret`

```
[=~"_bucket$"]: #S3Secret & {
    namespace: string
    if zone.product == "caascad" && zone.name != "corp-stg" {
        _vault_login: run.pre_tasks["vault_\(zone.name)"]
        url:          _vault_login.url
    }
    if zone.name == "corp-stg" {
        _vault_login: run.pre_tasks["vault_corp"]
        url:          _vault_login.url
    }
    path: *"secret/zones/\(zone.provider.type)/\(zone.name)/b
    if zone.product == "caascad" {
        path: "secret/bucket/\(namespace)"
    }
}
```

Exemple

```
let BucketUsages = run.pre_tasks.bucket_usages_read.data

// devient

let BucketUsages = run.pre_tasks.usages_read_bucket.data
```

Problème de concurence avec Cue Flow

```
❯ trackbone plan -c fe_bastion -n8 -t staging

panic: runtime error: invalid memory address or nil pointer d
[signal SIGSEGV: segmentation violation code=0x1 addr=0x8 pc=
```

bug déjà présent sur cue 0.4

Explication

Pour chaque tache, Cue Flow attribut un état, qui peut être `Waiting` -> `Ready` -> `Running` -> `Terminated`.

- Une tache qui dépend d’une autre est `Waiting`,
- Une tache qui n’a plus de dépendances (tous ses inputs sont concrets) est `Ready`

A chaque cycle Cue Flow

- écrit les outputs des taches `Terminated` dans le contexte Cue
- lit les inputs des taches `Ready` dans le même contexte Cue
- passe les taches `Ready` à `Running`

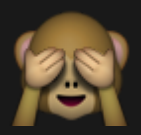
- Il crée 1 thread par tache `Running` (ce n’est pas configurable)
- Il écrit et lit dans le contexte Cue en même temps.

 Cue n’est pas thread safe 

Solution (de contournement)

J’ai ajouté un mutex dans trackbone, pour réduire le nombre d’accès parallèles au contexte.

Mais comme flow fait également des écritures dans son coin

J’ai ajouté des wait pour lui laisser le temps de faire sa tambouille 

Rappel sur la fermeture (close) des définitions

Quel est le résultat de ce code ?

```
package test

#Test: {
  a: string
}

#TEST2: {
 #Test
 b: string
}

#TEST3: #Test & {
  b: string
}
```

Réponse

```
#TEST3.b: field not allowed:
    -:5:8
    -:14:9
    -:15:3
```

Une structure est ouverte

On peut fermer une structure avec `close`

```
ouvert: {
  a: string
}

ferme: close({
  a: string
})
```

Une définition est fermée

Les préfixes # ou _# représente une définition qui est fermé

```
#Ferme: {
  a: string
}

#Ouvert: {
  a: string
  ...
}
```

On peut ouvrir une définition avec l'ellipse . . .

Fermer une structure

```
#Ferme: {
  a: string
}

ferme: #Ferme & {
  a: "ok 🤘"
}

ouvert: {
  #Ferme
  b: string
}
```

Problème du modulo

Changement de l’affichage des float

Précision à 1 chiffre après la virgule automatique

Modulo ne retourne pas un entier

```
let ThirdGroupByte = mod(div(RelativeClusterIndex, nbClusterF
let GroupCidr = "10.0.\(ThirdGroupByte).0/24"
```

Cue 0.4

GroupCidr = “10.0.128.0/24”

Cue 0.7

GroupCidr = “10.0.128.0.0/24”

Non ce n’est pas IPv5

Explication

- En Go: toutes les fonctions de Math retournent un float
- Donc `mod`, `div`, etc en Cue retourne un float

En cue 0.4

L’interpolation retour la valeur la plus courte

“2.5 -> 2.5”

“2.0 -> 2”

“0 -> 0”

En cue 0.7

L’interpolation retour la valeur avec une précision de 1 nombre après la virgule (*il me semble*)

“2.5 -> 2.5”

“2.0 -> 2.0”

“0 -> 0.0”

Solution

On fait la conversion en string manuellement

```
let ThirdGroupByte = mod(div(RelativeClusterIndex, nbClusterF
let Str = strconv.FormatFloat(ThirdGroupByte, 102, 0, 64)
let GroupCidr = "10.0.\(Str).0/24"
```

Remarque:

```
la syntaxe normale c'est
strconv.FormatFloat(ThirdGroupByte, 'f', 0, 64)

mais le paramètre 'f' est transformé en bytes
au lieu de int à cause de cue.
```

Conclusion / RAF

- Une régression peut en cacher une autre 
- Je recommande l’abandon de Cue Flow
- Garder Cue pour
  - la validation de la configuration
  - la génération des inputs pour les jobs
  - le calcul des dépendances explicitement

RAF

- Finalisation des tests
- Upgrade de env-ng et trackbone en simultanée bientôt 
- Traitements parallélisés par zone pour réduire la conso mémoire