

Concourse

Cloud-Native CI/CD

Devoxx France

Paris

18 / 04 / 2018

Daniel Garnier-Moiroux

Software Engineer @ Pivotal Labs

dgarnier@pivotal.io

[@Kehrlann](#)

[Kehrlann @ github](#)

Ca peut aller un peu vite

On aura du temps pour les questions

Et le talk est enregistré



Au programme aujourd'hui

- Principes fondamentaux
- Concepts et fonctionnement
 - Resources, jobs, pipelines
 - GUI immutable, outil CLI *fly*
- Démo de l'outil avec un mini pipeline
- Ton premier (vrai) pipeline



Principles

Concourse

- Quelques principes fondamentaux
 - **Containers first** : les jobs tournent dans des containers, stateless
 - **Stateless workers** : des *resources* externes pour le state
 - **Pipeline first** : Concourse a été fait pour des pipelines
 - **Configuration as code** : toute la config est dans des fichiers YAML



Sous le capot ...

Resource

- Permet de stocker les state entre deux jobs du pipeline
- De nombreuses resources supportées :
 - git
 - s3
 - time
 - semver
 - cf
 - ...
 - Plein de resources community ! (github, slack, jira, sonar ...)
- Brique de base, que Concourse peut:
 - *check*, pour découvrir des nouvelles versions
 - *get*, pour l'utiliser dans un job/task
 - *put*, pour y envoyer le résultat d'un job/task
- Simple à implémenter: trois scripts (max) dans un container de votre choix

Exemple de resource : git

- Définition :

```
resources:  
- name: backend-repo  
  type: git  
  source:  
    uri: git@github.com/Kehrlann/concourse-demo.git  
    branch: master  
    private_key: ...  
    paths: [ci, backend]
```

- Utilisation :

```
jobs:  
- name: build-backend  
  plan:  
    - get: repo  
      resource: backend-repo  
      trigger: true  
    - ... [other tasks] ...
```

Jobs

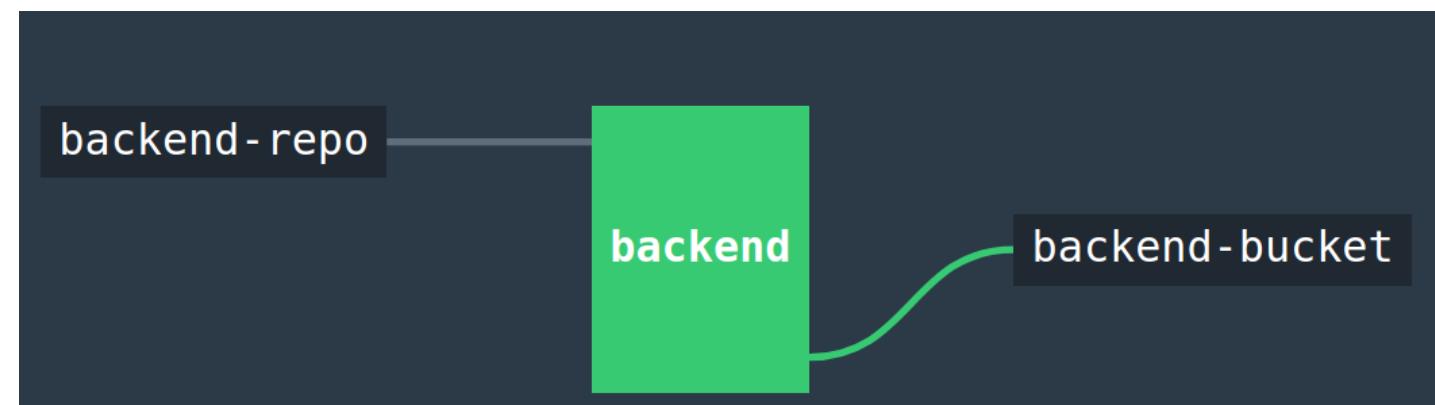
- Deuxième brique de base
- Série de steps:
 - *task*, pour exécuter du code arbitraire dans un container
 - *get*, pour récupérer des resources à utiliser en input
 - Peut entraîner l'exécution du job ou non, via un flag *trigger*
 - *put*, pour pousser des outputs de jobs vers des resources

Exemple de job

Dans le code

```
jobs:  
- name: build-backend  
  plan:  
    - get: repo  
      resource: backend-repo  
      trigger: true  
    - task: build  
      file: repo/ci/tasks/build-backend.yml  
    - put: backend-bucket  
      path: output
```

Dans le GUI



Pipeline

- Assemblage de jobs qui tournent en parallèle ou séquentiellement
- Le lancement des jobs est contrôlé par les *resources* quand il y a une nouvelle version disponible
- Les pipelines sont construits autour des resources :
 - Le job A produit une resource 1
 - Le job B consomme la resource 1, si le job A a réussi
 - C'est un pipeline [job A] -> (res 1) -> [job B]
- Toute l'info est dans un fichier de configuration, e.g. pipeline.yml
- Exemple de pipeline

Concourse UI

- Le GUI est un site web, immutable
- Interaction via l'outil en CLI *fly*
 - *fly set-pipeline*, *fly trigger-job*, ..
 - *fly execute* pour envoyer une *task* à exécuter au worker
 - *fly intercept* pour exécuter un shell dans le container d'un job

Ca suffit les slides, on veut du code !

