Publik - Development #81739

passer de tox à nox (?)

28 septembre 2023 13:00 - Frédéric Péters

Statut:	Solution validée	**Début:**	28 septembre 2023
Priorité:	Normal	**Echéance:**	
Assigné à:		**% réalisé:**	0%
Catégorie:		**Temps estimé:**	0:00 heure
Version cible:			
Patch proposed:	Non	**Club:**	Non
Planning:	Non		

Description

Parce que #81735 a été posé, pour pouvoir discuter plus globalement :

On chercherait à améliorer quoi ?

Perso mon problème principal avec tox c'est dans les moments où il y a des instructions conflictuelles sur les versions, typiquement on met explicitement django>=4.2,<4.3 pour tester avec cette version de django mais parce que dans un setup.py du module ou d'une de ses dépendances il se trouve à finalement rester sur l'ancienne version et il faut penser à aller vérifier ce qu'il a installé comme versions pour se rendre compte du problème.

Demandes liées:

Lié à Lingo - Development #81735: Tester nox	**Solution proposée** 28 septembre 2023

Historique

#1 - 28 septembre 2023 13:00 - Frédéric Péters

- Lié à Development #81735: Tester nox ajouté

#2 - 28 septembre 2023 15:16 - Corentin Séchet

Frédéric Péters a écrit :

> Perso mon problème principal avec tox c'est dans les moments où il y a des instructions conflictuelles sur les versions, typiquement on met explicitement django>=4.2,<4.3 pour tester avec cette version de django mais parce que dans un setup.py du module ou d'une de ses dépendances il se trouve à finalement rester sur l'ancienne version et il faut penser à aller vérifier ce qu'il a installé comme versions pour se rendre compte du problème.

Je sais pas si j'ai bien compris. J'ai testé en local, si on fait un session.install('-e', '.', 'django>3.4') par exemple, dans nox, ça prend en compte à la fois les contraintes de versions du setup.py et de celles qui sont passées à nox. Donc ça permet d'ajouter des contraintes de version dans noxfile.py par dessus celles du setup.py (et de lever une erreur s'il y aucune version ne satisfait les deux).

Mon problème avec tox, c'est qu'il est très rigide. Par exemple il s'attend à avoir une liste de dépendances qui sont des packages pythons, ce qui oblige à bricoler avec des scripts bash pour installer un environnement node pour les tests JS. Il installe aussi systématiquement le projet en cours dans l'environnement, alors que j'en ai pas besoin (et du coup ça prend beaucoup plus de temps que ça devrait). La config de nox se fait en python, c'est plus flexible et ça me permet d'installer un environnement avec juste node et ce qu'il faut pour les tests JS.

De manière plus générale, dès que j'ai eu à configurer un truc dans tox, je me suis noyé dans la doc. Je trouve celle de nox beaucoup plus concise. Et le DSL dans un .ini je trouve ça très dur à lire, tandis que le python, ça parle à tout le monde. Le python permet aussi de factoriser des choses simplement avec des méthodes, par ex. les mêmes dépendances déclarées à deux endroits différents pour pylint ou les tests unitaires dans tox, ça se factorise très simplement dans nox.

#3 - 29 septembre 2023 10:45 - Frédéric Péters

> Je sais pas si j'ai bien compris. J'ai testé en local, si on fait un session.install('-e', '.', 'django>3.4') par exemple, dans nox, ça prend en compte à la fois les contraintes de versions du setup.py et de celles qui sont passées à nox. Donc ça permet d'ajouter des contraintes de version dans noxfile.py par dessus celles du setup.py (et de lever une erreur s'il y aucune version ne satisfait les deux).

Le problème que j'ai avec tox est sur des instructions contradictoires, j'ai posé https://jenkins.entrouvert.org/job/gitea/job/welco/job/wip%252Ftox-version/1/console en exemple, il y a là un environnement qui demande explicitement django 2.2 mais quand on regarde ce que ça installe, surprise, on a 3.2 :

```
10:31:02  py3-django22-drf312 installed: ...,Django==3.2.21,...
```

Ensuite le build échoue pour d'autres raisons mais parfois le build fonctionne très bien avec la version qui a été tirée et on ne va pas se rendre compte que l'environnement mis en place pour valider que ça va être ok avec telle version n'utilise en fait pas la version demandée.

~~

> ce qui oblige à bricoler avec des scripts bash

La démonstration de cet avantage pourrait passer par la suppression de session.run('bash', './getlasso3.sh', external=True) ?

~~

Je n'ai vraiment aucune objection à quitter tox, il y aurait par contre je trouve à tester nox sur une situation moins simple, typiquement la validation sur plusieurs versions de django, pour valider l'affaire mais aussi pour avoir un exemple de comment ça se ferait (parce qu'avec le temps je peux ajouter un environnement au tox.ini pour arriver à faire ça, mais sur le noxfile.py je ne saurais évidemment pas faire sans aller aujourd'hui lire la documentation).

Aussi la PR lingo conserve un bout avec tox, ce qui est moyen mais j'imagine c'est parce que la version packagée debian est jugée trop ancienne et qu'il faut d'une manière ou d'une autre bootstrapper ça ?

#4 - 02 octobre 2023 10:40 - Corentin Séchet

Frédéric Péters a écrit :

> Le problème que j'ai avec tox est sur des instructions contradictoires, j'ai posé https://jenkins.entrouvert.org/job/gitea/job/welco/job/wip%252Ftox-version/1/console en exemple, il y a là un environnement qui demande explicitement django 2.2 mais quand on regarde ce que ça installe, surprise, on a 3.2 :

Je vais regarder en détail et m'assurer que ça serait corrigé par Nox.

> > ce qui oblige à bricoler avec des scripts bash
>
> La démonstration de cet avantage pourrait passer par la suppression de session.run('bash', './getlasso3.sh', external=True) ?

Ok.

> Je n'ai vraiment aucune objection à quitter tox, il y aurait par contre je trouve à tester nox sur une situation moins simple, typiquement la validation sur plusieurs versions de django, pour valider l'affaire mais aussi pour avoir un exemple de comment ça se ferait (parce qu'avec le temps je peux ajouter un environnement au tox.ini pour arriver à faire ça, mais sur le noxfile.py je ne saurais évidemment pas faire sans aller aujourd'hui lire la documentation).

Ok

> Aussi la PR lingo conserve un bout avec tox, ce qui est moyen mais j'imagine c'est parce que la version packagée debian est jugée trop ancienne et qu'il faut d'une manière ou d'une autre bootstrapper ça ?

Non, j'ai fait attention de faire quelque chose qui fonctionne avec la version de Nox packagée dans la version de Debian sur laquelle tourne le worker Jenkins. J'ai juste fait ça parce que je ne sais pas comment y installer un paquet.

#7 - 03 octobre 2023 12:44 - Frédéric Péters

Réflexion supplémentaire, parce qu'avec nox ça doit devenir possible, arriver à un module commun pour certaines choses, pour par exemple ne pas avoir à aller modifier 20 modules pour y mettre (astroid<3 pylint<3) (#81905).

#8 - 03 octobre 2023 13:28 - Corentin Séchet

Sur https://git.entrouvert.org/entrouvert/lingo/pulls/113 :

- Pour la démonstration, j'ai ajouté une paramétrisation pour lancer les tests sur Django 3.1 & 3.2. Ça illustre le fait que le souci de mauvaise version installée serait résolu, puisque l'installation en 3.1 lève une erreur, aucune version ne pouvant satisfaire ce qu'on demande à pip.
- Utilisé les posargs pour lancer les tests / pylint uniquement sur certains fichier, et activer le coverage (ce qui permet d'utiliser le même venv pour les tests avec ou sans coverage)
- Déplacé getlasso3.sh et pylint.sh dans le noxfile. Pour pylint, il est possible de faire quelque chose de moins convolué en utilisant une version de nox plus récente qui permet de rediriger la sortie d'un session.run (C.F le code avant le commit "fix output pipe not available on Jenkin's nox version").
- Utilisé session.interactive pour déterminer si on est sur Jenkins ou pas (en local, user --non-interactive permet de simuler le fonctionnement sur Jenkins)
- J'ai eu l'erreur avec la nouvelle version de pylint évoqué dans #81905, j'ai downgradé la version à installer mais ça lève des faux positifs

Réflexion supplémentaire, parce qu'avec nox ça doit devenir possible, arriver à un module commun pour certaines choses, pour par exemple ne pas avoir à aller modifier 20 modules pour y mettre (astroid<3 pylint<3) (#81905).

Oui, on pourrait installer un package python en local & sur Jenkins, ou demander à nox de cloner un repo puis de l'importer, ou aller chercher un fichier de config partagé quelque part...

#9 - 27 novembre 2023 10:54 - Benjamin Dauvergne

- Tracker changé de Support à Development

- Statut changé de Nouveau à Solution validée

Go.