Oxidized

Une alternative viable à Rancid

S'oxyder est toujours mieux que de se rancir

Cyril LAVIER
Administrateur Système
webedia.

OXIDIZED

- Alternative à RANCID.
  - But originel : Remplacer RANCID pour JunOS et IOS.
  - Supporte maintenant une soixantaine d'OS/constructeurs.

- Propose une API RESTful et une CLI.

- Divers choix de backends tant pour la source que pour la destination, tous étant dépendants de la bibliothèque Ruby Oxidized.

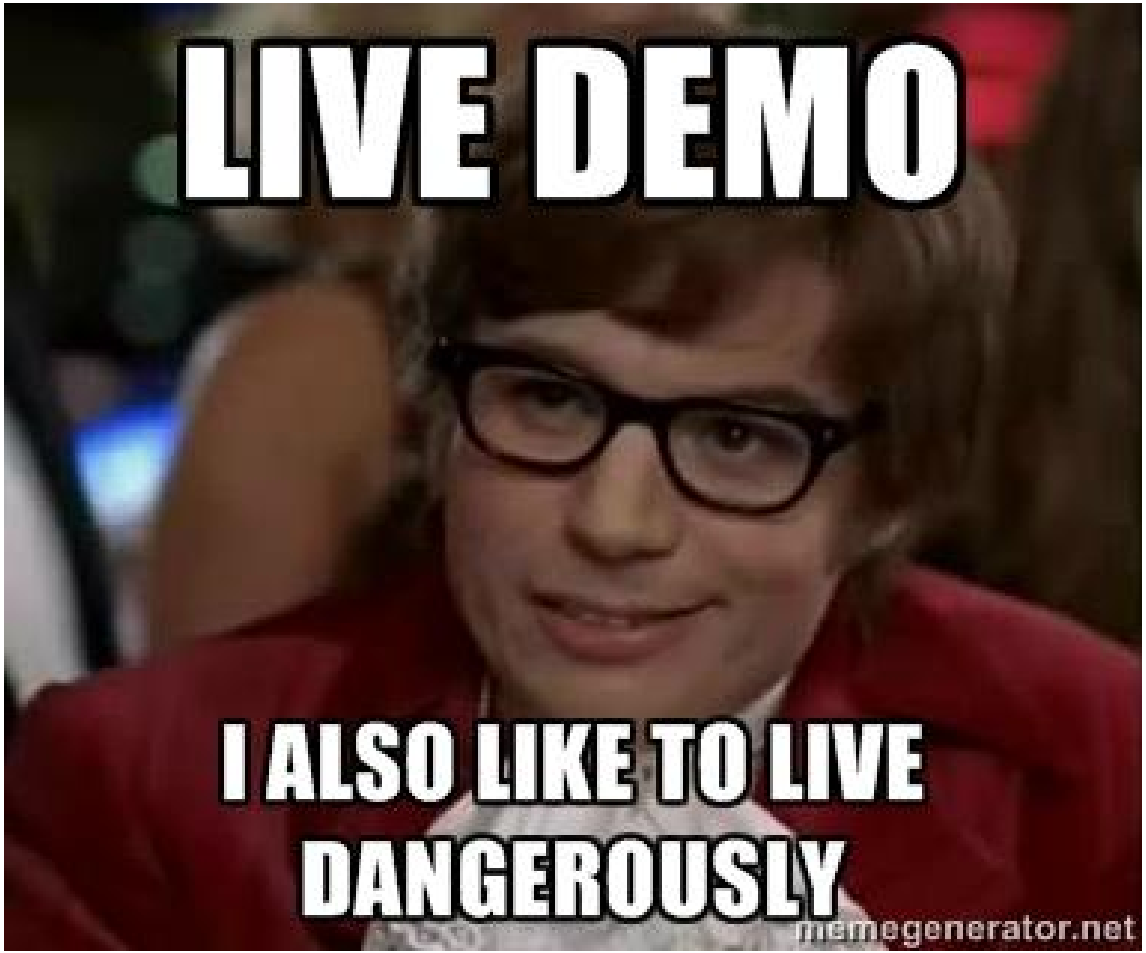
- Bibliothèque Ruby centrée autour de 4 objets :
  - Input : Méthodes pour aller récupérer la configuration des équipements
    - SSH, FTP, TFTP et Telnet.
  - Model : Utilisé pour les modules constructeurs.
    - JunOS, APC AOS, Cisco IOS, PFSense, etc...
  - Output : Utilisé pour stocker les configurations récupérées.
    - GIT, HTTP et File.
  - Source : Liste des équipements à versionner/sauvegarder.
    - CSV, HTTP et SQL.

L'interface WEB et la CLI

- Oxidized-web :
  - Permet d'aider à la migration depuis RANCID avec l'outil de migration :
    - Il lui faut le router.db, le cloginrc et il va générer un fichier CSV.
  - Vue synthétique sur tous les équipements gérés :
    - Diverses statistiques d'accès.
    - Moteur de recherche, classement par groupes ou type d'équipements.
    - Accès à l'historique des configurations (uniquement avec stockage GIT).
    - Téléchargement de la dernière configuration.
  - Un peu d'administration :
    - Reload la liste des équipements.
    - Forcer le poll d'un équipement en particulier.
  - Accès à l'API RESTful :
    - En dehors de l'outil de migration, tout est aussi accessible en JSON… y compris les configurations si le parseur Ruby n'explose pas en plein vol.

- Oxidized-script :
  - Utilise la lib Oxidized pour se connecter aux équipements et lancer des commandes/scripts.
  - Possibilité de lancer ces commandes sur des groupes entiers, et de les threader.
  - Propose une lib pour l'appeler nativement en Ruby.

Démo !!!!


LIVE DEMO
I ALSO LIKE TO LIVE DANGEROUSLY
memegenerator.net

Conclusion et liens utiles

- Oxidized est encore un projet jeune (créé en Avril 2013), mais il progresse et s'améliore régulièrement.
- La CLI et l'interface web peuvent faire gadgets au début, mais on se retrouve à les utiliser plus souvent qu'on ne pourrait y penser.
    - L'export en JSON permet d'utiliser Oxidized comme base pour des dashboards par exemple.
- Il peut se baser sur n'importe quel outil de gestion de parc qui utilise une BDD SQL.
- Il apporte des fonctionnalités supplémentaires à RANCID qui rendent l'exploitation de l'outil plus souple et simple.
- Liens des différents projets:
    - Oxidized : https://github.com/ytti/oxidized
    - Oxidized-web : https://github.com/ytti/oxidized-web
    - Oxidized-script : https://github.com/ytti/oxidized-script

Questions ?

Pas d'indice en bas de votre écran

webedia.

SLIDE BONUS

webedia.

Les modules de modèles constructeurs

- 4 mots clés :
  - cmd : Fichier de configuration ou commande permettant d'afficher la configuration.
  - prompt : Le prompt à matcher si "cmd" est une commande.
  - comment : Caractère utilisé pour les commentaires dans la configuration.
  - cfg : Une instance du module "input" utilisé, spécifie la connexion à l'équipement.

- Exemple avec le module APC_AOS :

```
class Apc_aos < Oxidized::Model
  cmd 'config.ini' do |cfg|
    cfg.gsub! /^; Configuration file\, generated on.*/, ''
  end
  cfg :ftp do
  end
end
```

 Déclaration du module
 Définition du fichier de configuration et éventuellement des actions à réaliser avant de stocker le fichier.
 Définition du protocole pour se connecter à l'équipement et éventuellement des options.

- À noter :
  - Il est possible de supporter plusieurs protocoles, dans ce cas, plusieurs modules peuvent être spécifiés à "cfg…. do" ou plusieurs blocs "cfg…. do" seront nécessaires.
  - Il doit y avoir autant de "cmd" que de commandes/fichiers à lancer/récupérer. L'output étant concaténé pour créer l'élément stocké.