Fabric

Jesús Espino García

Python-ESP-Centro

8 de Marzo de 2011

¿Qué es Fabric?

- Herramienta de despliegue.
- Herramienta ejecución de tareas.
- Herramienta para ejecución remota.

¿Para qué sirve Fabric?

- Despliegue de aplicaciones en entornos complejos.
- Despliegue de aplicaciones en entornos distribuidos.
- Ejecución de tareas administrativas habituales.
- Ejecución de tareas administrativas en entornos distribuidos.
- Scripts de construcción.
- ...

¿Por qué usar Fabric?

- Nos evita errores.
- Nos evita inconsistencias.
- Nos facilita las tareas periódicas.
- Disminuye los tiempos despliegue.
- Es Python!!

¿Como funciona Fabric?

- Se escribe un fichero `fabfile.py`.
- En este fichero se definen una serie de funciones (tareas).
- Se ejecuta la aplicación `fab <tarea>`.

Entorno

- La ejecución de fabric depende del entorno definido.
- El entorno es un objeto diccionario.
- Para cambiar el comportamiento modificamos el entorno.
- Cambios más habituales:
  - hosts: Host en los que ejecutar las tareas.
  - user: Usuario para las conexiones SSH
  - password: Password para las conexiones SSH
  - warn_only: No detener en caso de error
  - roledefs: Definición de roles (grupos de hosts)
  - roles: Roles sobre los que ejecutar las tareas.
  - fabfile: Fichero a ejecutar en vez de fabfile.py.

fabricrc

- El fichero ~/.fabricrc nos permite configuraciones generales.
- Este fichero afecta a todas las ejecuciones de fabric.
- Contiene un serie de parejas clave/valor.
- Las parejas clave valor rellenan el entorno.

Modelo de ejecución

- Construcción de una lista de tareas.
- Construcción de una lista de hosts.
- Se recorre la lista de tareas.
- Para cada hosts se ejecuta la tarea actual.

Lista de tareas

- Fabric extrae las tareas del fichero `fabfile.py` (Normalmente).
- Considera tarea todo objeto "Callable" que haya al importar el `fabfile.py`.
- No se consideran tareas las funciones/métodos del propio fabric.
- No se consideran tareas los objetos que su nombre empieza por _.
- Se pueden listar las tareas ejecutando `fab --list`.

Lista de hosts

- La lista de hosts se extrae del entorno.
- Si la tarea tiene un decorador de hosts sobreescribe los hosts del entorno.
- Si se le pasa como parámetro los hosts sobreescribe los hosts del entorno y decorador.
- Se extrae la lista de roles del entorno.
- Si la tarea tiene un decorador de roles se sobreescriben los roles del entorno.
- Si se le pasa como parámetro los roles sobreescribe los roles del entorno y decorador.
- Se hace la union de los hosts y de los hosts contenidos en los roles.

Tareas con parámetros

- Las tareas pueden recibir parámetros.
- El paso de parámetros sigue el siguiente modelos
  `fab <tarea>:<parm1>,<parm2>,<parm_name>=<parm_value>`
- El parámetro hosts permite definir los hosts sobre los que ejecutar.
- Si se encuentra el parámetro hosts se extrae de la lista de parámetros de la tarea.

Core

- colors (blue, cyan, green, magenta, red, white, yellow)
- context_managers (cd, hide, lcd, path, prefix, settings, show)
- decorators (hosts, roles, runs_once)
- network (disconnect_all)
- operations (get, open_shell, put, reboot, run, sudo, local, prompt, require)
- utils (abort, fastprint, indent, puts, warn)

Contrib

- contrib.console (confirm)
- contrib.django (project, settings_module)
- contrib.files (append, comment, contains, exists, first, sed, uncomment, upload_template)
- contrib.project (rsync_project, upload_project)

Ejemplos

- Despliegue de aplicación web distribuida.
- Despliegue de otra aplicación web distribuida.
- Despliegue de aplicación web compleja.
- Constructor de software.
- Script de administración.

Dudas

. . .