HERENCIA DE MANIFESTS

Para preparar nuestra imagen de aulas, la que vamos a subir a OpenGnsys, vamos a crear un manifest donde vamos a incluir todas las tareas que queremos que realice el equipo con el que vamos a realizar la imagen de aulas.

Dicho manifest puede llamarse, por ejemplo:

imagen_manifest.yaml

También quiero definir un catalog que voy a asociar a este manifest, por ejemplo:

imagen_catalog.yaml

No me olvido del catálogo general de apps que ya tenía:

catalog1.yaml

1 - ¿Cómo sería el manifest de un equipo con el que voy a realizar una imagen?

Si quiero realizar tareas previas a subir la imagen en ese equipo, puedo incluirle este manifest, ¿no?

Si mi equipo cliente tiene asociado el siguiente manifest: cliente1_manifest, podría hacer esto:

```
---
name: cliente1_manifest
managed_installs:
managed_uninstalls:
managed_updates:
included_manifests:
  - imagen_manifest
catalogs:
  - catalog1
```

Donde **imagen_manifest** se define como:

```
---
name: imagen_manifest
managed_installs:
managed_uninstalls:
managed_updates:
included_manifests:
catalogs:
  - imagen_catalog
```

2 - ¿Una vez que he terminado la imagen y antes de subirla al servidor…cómo debería dejar el manifest?

Como yo deje el manifest en el equipo de imagen -> es como se lo van a encontrar los equipos del aula tras la restauración.

Pero tras restaurar yo no quiero realizar las tareas que están en **imagen_manifest** (ya están hechas).

¡Podríamos hacer otro manifest para que se use en producción!

produccion_manifest.yaml

Y de nuevo asociar un catalog este manifest, por ejemplo:

producción_catalog.yaml

Justo antes de subir la imagen, debería dejar el manifest del equipo imagen tal que así:

```
---
name: cliente1_manifest
managed_installs:
managed_uninstalls:
managed_updates:
included_manifests:
  - produccion_manifest
catalogs:
  - produccion_catalog
```

Donde **produccion_manifest** se define como:

```
---
name: produccion_manifest
managed_installs:
managed_uninstalls:
managed_updates:
included_manifests:
  - imagen_manifest
catalogs:
  - produccion_catalog
```

EJERCICIO

Suponemos que **imagen_manifest** (y su catálogo asociado), tiene 3 tareas llamadas (imagen_tarea1, imagen_tarea2 e imagen_tarea3) , que son las que vamos a aplicar sobre el equipo de imagen.

También suponemos que **produccion_manifest** (y su catálogo asociado) tiene otras 3 tareas (produccion_tarea1, produccion_tarea2, produccion_tarea3), que vamos a aplicar una vez RESTAURADA la imagen.

Se propone:

1- Definir cómo van a ser los manifests **cliente1_manifest** y **imagen_manifest** mientras estoy HACIENDO LA IMAGEN.

2- Definir cómo va a ser el manifest **cliente1_manifest** justo ANTES DE SUBIR LA IMAGEN.

3- Definir cómo van a ser los manifests **cliente1_manifest** y **produccion_manifest** justo DESPUES DE CLONAR UN EQUIPO.

NOTA:

Las tareas de los catálogos serán del tipo:

```
imagen_tarea1:
  diplay_name: imagen_tarea1

  check:
    script: |
      Write-Host "imagen_tarea1"
      exit 0
  installer:
    location: /packages/imagen/imagen_tarea1.ps1
    hash:
AA4FFCFFCAB6859390E39352352E26F84DCB884EB61C19B73CE7C8AF4E0
55647
    type: ps1
  version: 1.0
```

Replicar la tarea de arriba para que **imagen_catalog** contenga:

- imagen_tarea1
- imagen_tarea2
- imagen_tarea2

```
produccion_tarea3:
  diplay_name: produccion_tarea3
  check:
    script: |
      Write-Host "produccion_tarea3"
      exit 0
  installer:
    location: /packages/produccion/produccion_tarea3.ps1
    hash:
AA4FFCFFCAB6859390E39352352E26F84DCB884EB61C19B73CE7C8AF4E0
55647
    type: ps1
  version: 1.0
```

Replicar la tarea de arriba para que **produccion_catalog** contenga:

- produccion_tarea4
- produccion_tarea5
- produccion_tarea6