

Conversación con Gemini

Exported: 15/6/2026 9:11:06

Jordi H.

Has dicho quiero aprender a usar cloud-init. Todavía no me atrevo a hacer un deploy completo de mis bare metal. Ahora mismo tengo media docena de ordenadores en bare metal que uso para hacer maquetas. Me ocupa mucho tiempo configurarlas una a una. me conformaría con instalar el s.o. (ubuntu server) desde un p en y luego configurarle un fichero cloud init a cada una. como sería la cosa? cada vez que arrancan puedo modificar cosas solamente tocando el fichero? como sería el fichero la primera

Gemini

Entiendo perfectamente esa frustración. Configurar seis máquinas a mano es la definición de "trabajo repetitivo" que cloud-init viene a solucionar. Lo bueno es que no necesitas una infraestructura de nube compleja para empezar a usarlo; Ubuntu Server facilita mucho este proceso mediante **Autoinstall**. Aquí tienes el desglose de cómo funciona y cómo podrías implementar

1. ¿Cómo funciona la magia?

A diferencia de un script que ejecutas una vez, cloud-init es un servicio que se activa durante el arranque. Sin embargo, hay una distinción clave que debes conocer

- Fase de Instancia (First boot): La mayoría de las configuraciones (crear usuarios, instalar paquetes, particionar) ocurren solo la primera vez que el sistema arranca
- Arranques posteriores: En cada reinicio, cloud-init comprueba si hay cambios, pero por defecto no vuelve a ejecutar todo (para no machacar tus datos). Si quieres que algo se ejecute siempre, se usan módulos específicos como bootcm

2. El fichero inicial: `user-data`

Para Ubuntu Server, el archivo se llama habitualmente user-data. Debe empezar sí o sí con la línea `#cloud-config`. Aquí tienes un ejemplo básico para tus maquetas:YAM

CODE

```
#cloud-config
autoinstall:
  version: 1
  identity:
    hostname: maqueta-01
    password: "$6$examp$D5p98n9e9asda..." # Password hashed
    username: analista
  ssh:
    install-server: true
    authorized-keys:
      - ssh-rsa AAAAB3Nza... (tu clave pública)
  packages:
    - python3-pip
    - git
    - curl
  user-data:
    package_upgrade: true
  runcmd:
    - [ echo, "Maqueta lista para analítica", ">", /etc/motd ]
```

3. ¿Cómo se lo pasas al Bare Metal?

Como vas a usar un pendrive, tienes dos caminos principales para que el instalador de Ubuntu (Subiquity) detecte tu configuración

Opción A: El método del "Pendrive con nombre" (Recomendado)

1. Formatea un segundo pendrive (o una partición del primero) con el nombre de etiqueta CIDATA.
2. Copia dentro dos archivos:
 - user-data: El código YAML de arriba.
 - meta-data: Un archivo vacío (solo para que no de error).
3. Pincha el instalador de Ubuntu y el pendrive CIDATA. El instalador leerá el archivo y hará todo solo.

Opción B: Modificar el GRUB

Si no quieres usar dos USBs, puedes editar la línea de arranque del instalador (pulsando e en el menú de inicio) y añadir

autoinstall ds=nocloud-net;s=/media/tu-usb/

4. ¿Puedo modificar cosas solo tocando el fichero?

Sí y no. * Durante la instalación: Sí. Cada vez que reinstales desde el USB, tomará los cambios que hayas hecho en el user-data

- Con la máquina ya encendida: No basta con cambiar el archivo en el USB. Cloud-init guarda una caché en /var/lib/cloud/. Si quieres "forzar" a que la máquina vuelva a leer el archivo como si fuera la primera vez, tendrías que ejecutar

`sudo cloud-init clean --reboot` Esto es útil si estás probando configuraciones de software, pero ten cuidado porque podría intentar recrear usuarios o configuraciones de red

Resumen de flujo para tus maquetas

1. Preparas el YAML: Decides qué paquetes de Python y qué usuarios quieres.
2. Lanzas el despliegue: Arrancas el bare metal con el USB de Ubuntu + el USB CIDATA.
3. Te vas a tomar un café: El instalador particiona, instala el SO, descarga Python, configura tu SSH y se reinicia solo
4. Acceso: Entrás directamente por SSH sin haber tocado el teclado de esa máquina.

¿Te gustaría que te ayude a generar el hash de la contraseña o a definir los paquetes específicos de Python que sueles usar para que ya queden preinstalados

Jordi H.

Has dicho toda esta gestión de hashes y authorized-keys, tiene alguna herramienta para facilitarlo? la gente de Canonical tiene algún producto? hay algo sencillo o me voy a un deployer con dhcp netboot y cosas así

Gemini

Es normal que te dé pereza andar lidiando con hashes de contraseñas y claves públicas en YAML. Para media docena de máquinas, saltar a un sistema de red completo (PXE/Netboot) puede ser como "matar moscas a cañonazos", pero hay puntos medios muy interesantes. Aquí tienes las opciones, desde lo más "manual" hasta el producto oficial de Canonical

1. El "atajo" de Canonical: MAAS (Metal as a Service)

MAAS es el producto estrella de Canonical para lo que buscas. Es básicamente "tu propia nube privada de bare metal"

- Cómo funciona: Instalas un "Controller" en una máquina. Esta toma el control del DHCP de tu red. Cuando conectas tus otros 6 ordenadores por red (PXE boot), MAAS los detecta, les hace un inventario de hardware y los deja en estado "Ready"
- La ventaja: Desde un panel web haces clic en "Deploy", eliges Ubuntu, pegas tu clave SSH y listo. Él se encarga de todo el cloud-init por detrás
- ¿Es para ti? Si planeas montar y desmontar maquetas con frecuencia, sí. Si solo quieres instalarlas una vez y olvidarte, quizá la curva de aprendizaje de configurar el DHCP y las redes de MAAS sea demasiado para 6 máquinas

2. Herramientas para facilitar el YAML (Sin desplegar servidores)

Si prefieres seguir con el método del USB pero odias el formato, existen utilidades que te ahorran el trabajo sucio

A. `ssh-import-id` (El truco maestro)

En lugar de pegar tu clave pública gigante en el YAML, cloud-init puede importarla directamente de GitHub o Launchpad

Si tu usuario de GitHub es `tu_usuario`, en el fichero pones:YAML

CODE

```
ssh_import_id:  
- gh:tu_usuario
```

Esto descargará automáticamente tus claves públicas de GitHub al instalarse. Es infinitamente más limpio.

B. Generador de contraseñas (Hash)

No pongas la contraseña en texto plano. Para generar el hash que entiende Ubuntu (SHA-512) sin volverte loco, usa este comando en cualquier terminal Linux:Bas

CODE

```
python3 -c 'import crypt; print(crypt.crypt("tu_contraseña_secreta",  
crypt.mksalt(crypt.METHOD_SHA512)))'
```

Copia ese chorizo de letras y números en el campo password: del YAML.

C. Autoinstall Generator (Web/Comunidad)

No hay una web "oficial" interactiva de Canonical que te escupa el YAML (increíble pero cierto), pero el instalador de Ubuntu tiene un truco: **si instalas una máquina a mano una última vez**, al terminar, busca el archivo e

`/var/log/installer/autoinstall-user-data`

Ese archivo es el "resumen" de lo que acabas de hacer, ya convertido a formato cloud-init. Puedes usarlo como plantilla para las otras 5

3. Mi recomendación para tu caso

Dado que estás centrado en **Python y analítica**, no pierdas dos días configurando un servidor MAAS o una red PXE compleja

1. Usa `ssh-import-id` para no tocar claves SSH.

2. Crea una sola plantilla user-data con el hash de tu contraseña y tus librerías de Python (pandas, numpy, scikit-learn) ya en la sección packages
3. Usa el método del USB CIDATA. Es lo más "portable" y menos intrusivo para una red doméstica o de oficina pequeña

Un último consejo pro: Si vas a cambiar mucho la configuración de Python, no la metas toda en el cloud-init. Usa cloud-init solo para dejar el SO base, el usuario y SSH, y luego usa **Ansible** (que es Python puro) desde tu PC principal para gestionar las 6 máquinas a la vez. ¿Te suena Ansible o prefieres que todo, absolutamente todo, se quede configurado desde el primer arranque del U

Jordi H.

Has dicho ya me gusta la idea de partir el trabajo entre cloud-init y ansible. me parece una buena inversión de futuro. me gustaría profundizar en maas. necesito contratar servicios profesionales de canonical o es de uso libre? me puedes dar más pinceladas de cómo se us

Gemini

¡Excelente decisión! Pasar de "configurar a mano" a un flujo de **MAAS + Ansible** es subir a la liga profesional de la infraestructura. Es exactamente como trabajan los centros de datos modernos, pero a escala de tu laboratorio. Aquí tienes los detalles sobre MAAS en 202

1. ¿Es de uso libre o hay que pagar?

La respuesta corta es: **Es de uso libre (Open Source)**. Canonical ofrece MAAS bajo una licencia que permite instalarlo y gestionar tantas máquinas como quieras sin pagar licencias por nodo. Lo que ellos venden es el soporte profesional y funciones avanzadas para empresas (como la Alta Disponibilidad del controlador o el RBAC detallado), pero para tu laboratorio de 6 máquinas, la versión **Self-support (gratuita)** es más que suficiente y no tiene límites de funcionalidad bás

2. ¿Cómo funciona MAAS? (El flujo de trabajo)

Imagina que MAAS convierte tus ordenadores físicos en "instancias" de una nube privada. El proceso se divide en estas fases

1. Instalación del Controlador: Instalas Ubuntu Server en una máquina (puede ser una de las 6 o una pequeña aparte) y ejecutas `sudo snap install maas`. Este equipo será el "cerebro"
2. Enlistment (Detección): Conectas tus otros ordenadores por Ethernet. Al encenderlos, deben intentar arrancar por red (PXE Boot). MAAS los detectará automáticamente, registrará su CPU, RAM y discos, y los apagará. Ahora aparecen en tu panel web
3. Commissioning (Pruebas): Le pides a MAAS que "comisione" las máquinas. Las encenderá, ejecutará tests de estrés y verificará que todo el hardware funciona bien
4. Ready (Listas): Las máquinas se quedan en estado "Ready" (apagadas).
5. Deploy (Despliegue): Cuando necesitas una maqueta, vas al panel de MAAS, seleccionas la máquina y pulsas "Deploy". MAAS la enciende, le instala Ubuntu (o la distro que quieras) y le inyecta el cloud-init que tú hayas configurad

3. ¿Cómo encaja con tu analítica y Ansible?

Aquí es donde se pone interesante para tu perfil:

- Integración con Ansible: MAAS tiene un inventario dinámico. En lugar de escribir las IPs de tus 6 máquinas en Ansible a mano, Ansible le pregunta a MAAS: "¿Qué máquinas tienes encendidas?" y MAAS le responde con las IPs actuales
- Gestión de Red: MAAS gestiona el DNS y el DHCP. Si una máquina se llama maqueta-python-01, podrás

- hacerle ping maqueta-python-01.maas desde cualquier lado sin configurar nada más
- Composición de Scripts: Puedes subir tus scripts de Python como "Curtain scripts" o usarlos en la fase de user-data de cloud-init para que, al terminar el despliegue, la máquina ya tenga instalado Jupyter, Pandas o tus entornos de Cond

4. ¿Qué necesitas para empezar hoy?

Para que MAAS funcione bien en tu entorno de "bare metal", necesitas dos cosas críticas:

1. Acceso al BIOS/UEFI: Debes entrar en cada una de tus 6 máquinas y activar el "Boot from Network" (PXE) como primera opción de arranque
2. Control del DHCP: MAAS prefiere gestionar el DHCP de la red. Si tu router de casa/oficina ya da IPs, podrías tener conflictos. Lo ideal es que MAAS gestione una subred pequeña solo para esas máquinas

Tip de experto: Si tus máquinas tienen **IPMI** (como servidores Dell iDRAC o HP iLO), MAAS puede encenderlas y apagarlas él solo. Si son PCs normales, tendrás que encenderlas tú a mano la primera vez o usar **Wake-on-LAN**. ¿Tienes un ordenador extra para que haga de "servidor MAAS" o tendrías que sacrificar uno de tus 6 nodos para es

Jordi H.

Has dicho tengo un ordenador con dos nic. una se conecta a la red corporativa y la otra está en una red aislada en un switch donde se conectan las otras cinco, que solamnte tienen una nic. ya tengo configurado e l ordenador con dos nic para que haga nat y actualmente los otros nodos ya ven la red corporativa (y internet) y, por ejemplo, se actualizan muy bien. A la inversa, no tengo nada declarado que los haga visibles hacia afu

Gemini

Esa estructura de red que tienes es de "libro de texto" para montar un entorno de **MAAS** profesional en casa o en la oficina. Tienes lo que se llama un **Region/Rack Controller** con doble interfaz, lo cual es la configuración recomendada por Canonical para separar el tráfico de gestión del tráfico corporativo. Aquí tienes cómo encaja MAAS en tu topología actu

1. El rol de tu ordenador con dos NICs

Este equipo se convertirá en tu **MAAS Controller**.

- NIC 1 (Corporativa/Internet): Se queda como está. MAAS la usará para descargar las imágenes de Ubuntu, actualizaciones y paquetes de Python que necesites
- NIC 2 (Switch aislado con 5 nodos): Aquí es donde ocurre la magia. MAAS se convertirá en el "dueño" de esta red

2. Configuración del DHCP en MAAS

Como ya tienes el NAT configurado, el paso más importante es decirle a MAAS que **él debe gestionar el DHCP** en la interfaz de la red aislada (NIC 2)

- MAAS proporcionará las IPs a los otros 5 nodos.
- Al ser el servidor DHCP, cuando un nodo arranque por red (PXE), MAAS le dirá: "Toma esta IP y ahora descarga este micro-sistema operativo para que yo pueda inventariar tu hardware"

3. El proceso de "Enlistment" y "Commissioning"

Con MAAS instalado en el equipo principal:

1. Enciendes los 5 nodos y te aseguras de que en la BIOS el PXE Boot esté el primero.
2. Los nodos pedirán IP a tu NIC 2. MAAS los verá y aparecerán en el panel web

(http://localhost:5240/MAAS)

3. En ese momento, los nodos están "registrados" pero no configurados.
4. Lanzas el Commissioning: Los nodos arrancan un Ubuntu efímero en RAM, chequean que los discos y la CPU están bien, y se apagan

4. Despliegue con Cloud-Init personalizado

Ahora que MAAS conoce tus 5 máquinas, puedes desplegarlas. MAAS te permite subir archivos de **cloud-config** (user-data) globales o por máquina

- Primera vez: Le das a "Deploy", seleccionas Ubuntu 24.04 y MAAS les inyecta el cloud-init para crear tu usuario de analista y las authorized-keys
- Visibilidad: Como bien dices, ahora no son visibles desde fuera. MAAS te permite configurar el DNS interno para que desde el Controller puedas hacer ssh nodo1.maas, ssh nodo2.maas, etc

5. Preparación para Ansible

Una vez que MAAS termina el despliegue (tarda unos 5-10 minutos por nodo), las máquinas están listas para recibir comandos. Aquí es donde entra tu interés por Python

1. Instalas Ansible en el ordenador de las dos NICs.
2. Usas el plugin de inventario de MAAS para Ansible.
3. Lanzas tus Playbooks de Ansible para instalar conda, pandas, scipy o lo que necesites en los 5 nodos simultáneamente

Un detalle técnico importante:

Puesto que tus nodos solo tienen una NIC y están en una red aislada, MAAS les configurará automáticamente como **Gateway** la IP de tu NIC 2. Como ya tienes el NAT funcionando, tus nodos de analítica tendrán salida a internet inmediata para bajar librerías de Python sin que tú tengas que tocar una sola tabla de rutas manualmente en ellos. **¿Te atreverías a instalar MAAS (es un simple `sudo snap install maas`) en el ordenador esclavos?***
