Logotipo

El contenido generado por IA puede ser incorrecto.

MODAM Herramientas de Automatización de Despliegues

Maestría en Desarrollo y Operaciones de Software

**Actividad grupal: Empleo de Chef Workstation**

Grupo: **1040**

Equipo: **1A**

Integrantes del equipo:

**Domínguez Urías, Didier**

**Lopez Flamenco, Gerardo**

**Nieto Rivera, Athena**

**Soto Audelo, Jorge**

**Trejo Figueroa, José**

Julio de 2025

Tabla de contenido

[Instalación y configuración de Chef Workstation 4](#_Toc203814130)

[Actualización del sistema 4](#_Toc203814131)

[Descarga del instalador de Chef Workstation 4](#_Toc203814132)

[Instalación del paquete descargado 5](#_Toc203814133)

[Verificación de la instalación 6](#_Toc203814134)

[Verificación del binario y estructura interna 6](#_Toc203814135)

[Generación de estructura base para trabajo con recetas 7](#_Toc203814136)

[Cookbooks 8](#_Toc203814137)

[Cookbook 1: fastapi\_app 8](#_Toc203814138)

[Paso 1: Crear el cookbook 8](#_Toc203814139)

[Paso 2: Crear el archivo de plantilla main.py 9](#_Toc203814140)

[Paso 3: Editar la receta default.rb 10](#_Toc203814141)

[Paso 4: Ejecución del cookbook y validación del despliegue 13](#_Toc203814142)

[Cookbook 2: Instalación de Ngrok 16](#_Toc203814143)

[Paso 1: Crear el cookbook 16](#_Toc203814144)

[Paso 2: Editar la receta default.rb 17](#_Toc203814145)

[Paso 3: Ejecutar el cookbook 17](#_Toc203814146)

[Paso 4: Lanzar el servidor local y script de Ngrok 17](#_Toc203814147)

[Pruebas automatizadas con InSpec 17](#_Toc203814148)

[Cookbook 3: Instalación y despliegue de Jenkins automatizado 20](#_Toc203814149)

[Paso 1: Crear el cookbook 20](#_Toc203814150)

[Paso 2: Editar la receta default.rb 20](#_Toc203814151)

[Paso 3: Ejecutar la receta 22](#_Toc203814152)

[Paso 4: Validar Jenkins en el navegador 23](#_Toc203814153)

[Cookbook 4: WordPress 24](#_Toc203814154)

[Paso 1: Crear el cookbook 24](#_Toc203814155)

[Paso 2: Editar la receta default.rb 25](#_Toc203814156)

[Paso 3: Ejecutar el cookbook 26](#_Toc203814157)

[Paso 4: Validación del despliegue 26](#_Toc203814158)

[Cookbook 5: RunSpringbootAPP 28](#_Toc203814159)

[Paso 1: Crear el cookbook 28](#_Toc203814160)

[Paso 2: Seleccionar el repositorio de ejemplo 28](#_Toc203814161)

[Paso 3: Editar la receta default.rb 29](#_Toc203814162)

[Paso 4: Ejecutar el cookbook 29](#_Toc203814163)

[Paso 5: Validación en el navegador 30](#_Toc203814164)

[Repositorio con los cookbooks 31](#_Toc203814165)

[Tabla de valoración 32](#_Toc203814166)

[Reflexión sobre los desafíos enfrentados 33](#_Toc203814167)

[Conclusiones 34](#_Toc203814168)

[Recomendaciones 35](#_Toc203814169)

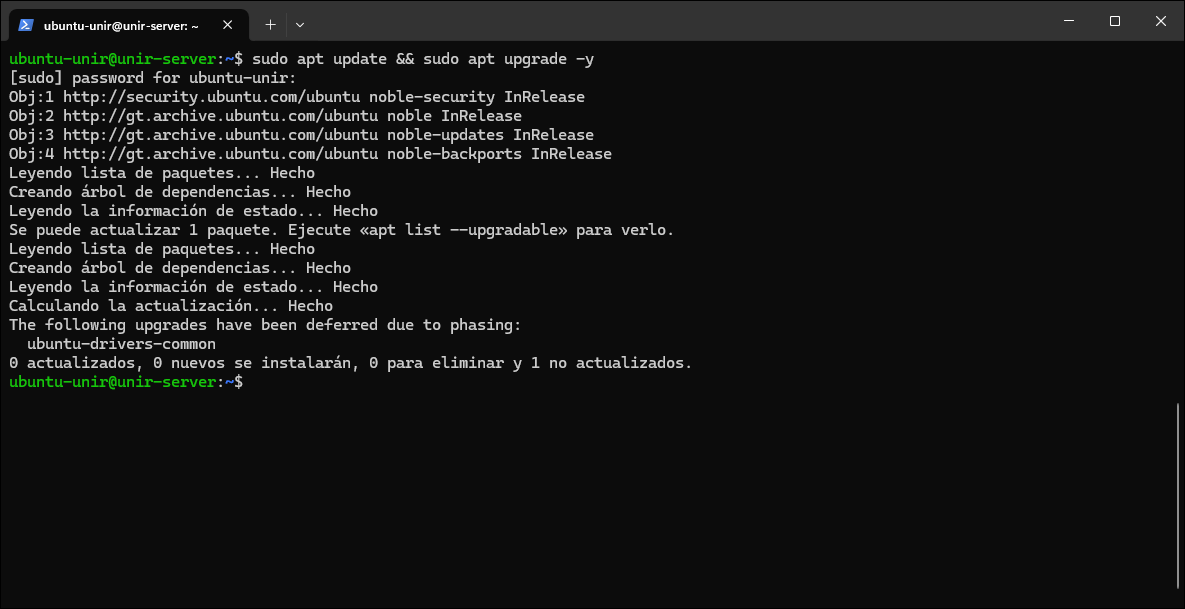
# Instalación y configuración de Chef Workstation

Para comenzar con la actividad, se procedió a realizar la instalación de Chef Workstation en un servidor con sistema operativo Ubuntu Server 24.04 LTS, siguiendo una serie de pasos sistemáticos que aseguran una instalación limpia, actualizada y funcional. A continuación, se detalla el proceso paso a paso:

## Actualización del sistema

Como buena práctica inicial, se actualizó el sistema para garantizar que todos los paquetes estén en su última versión disponible.

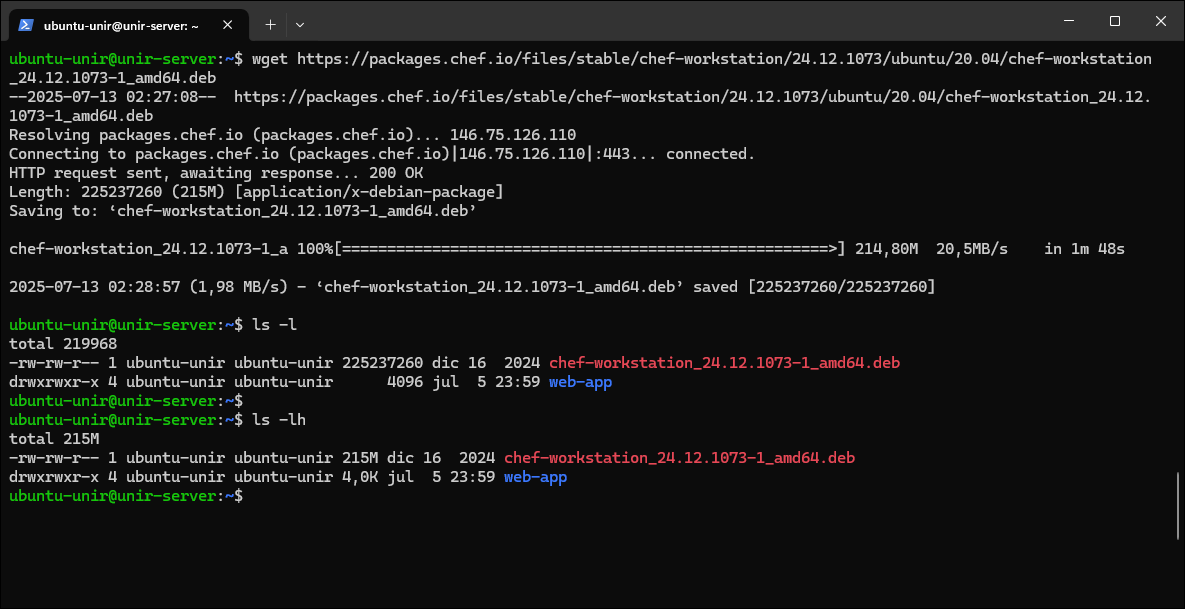
sudo apt update && sudo apt upgrade -y



## Descarga del instalador de Chef Workstation

Se descargó la versión Chef Workstation 24.12.1073, correspondiente a la LTS más reciente (julio 2025), desde el repositorio oficial de paquetes de Chef.

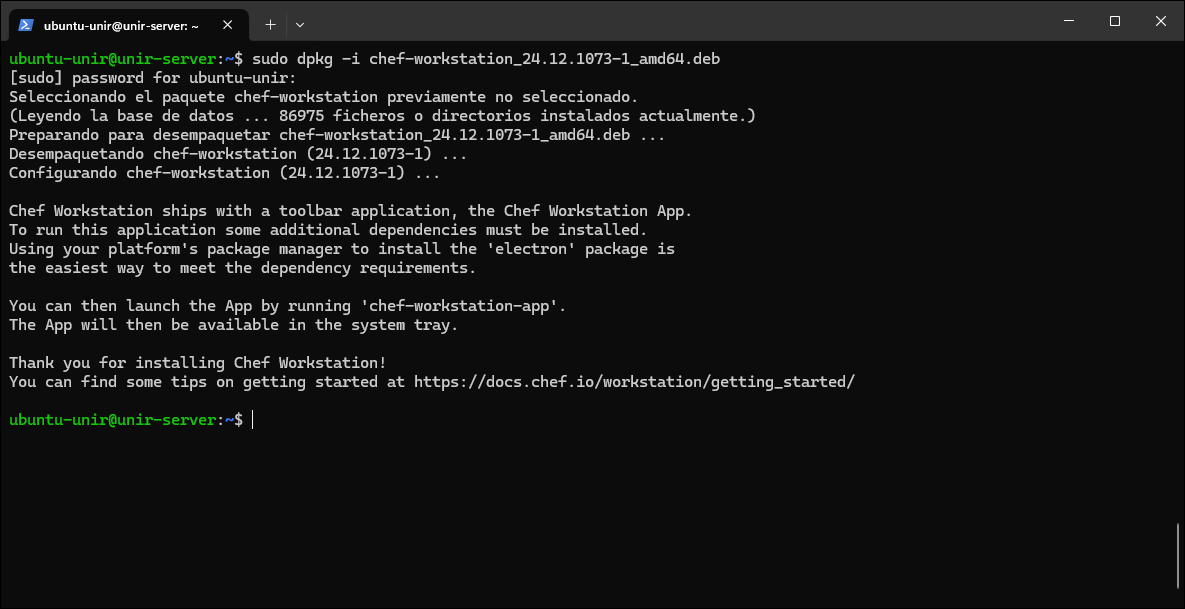
wget https://packages.chef.io/files/stable/chef- workstation/24.12.1073/ubuntu/20.04/chef-workstation\_24.12.1073-1\_amd64.deb



## Instalación del paquete descargado

Una vez completada la descarga, se procedió a instalar el paquete ***.deb*** utilizando ***dpkg***.

sudo dpkg -i chef-workstation\_24.12.1073-1\_amd64.deb



## Verificación de la instalación

Finalizada la instalación, se verificó que Chef Workstation se hubiera instalado correctamente, mediante el comando:

chef --version



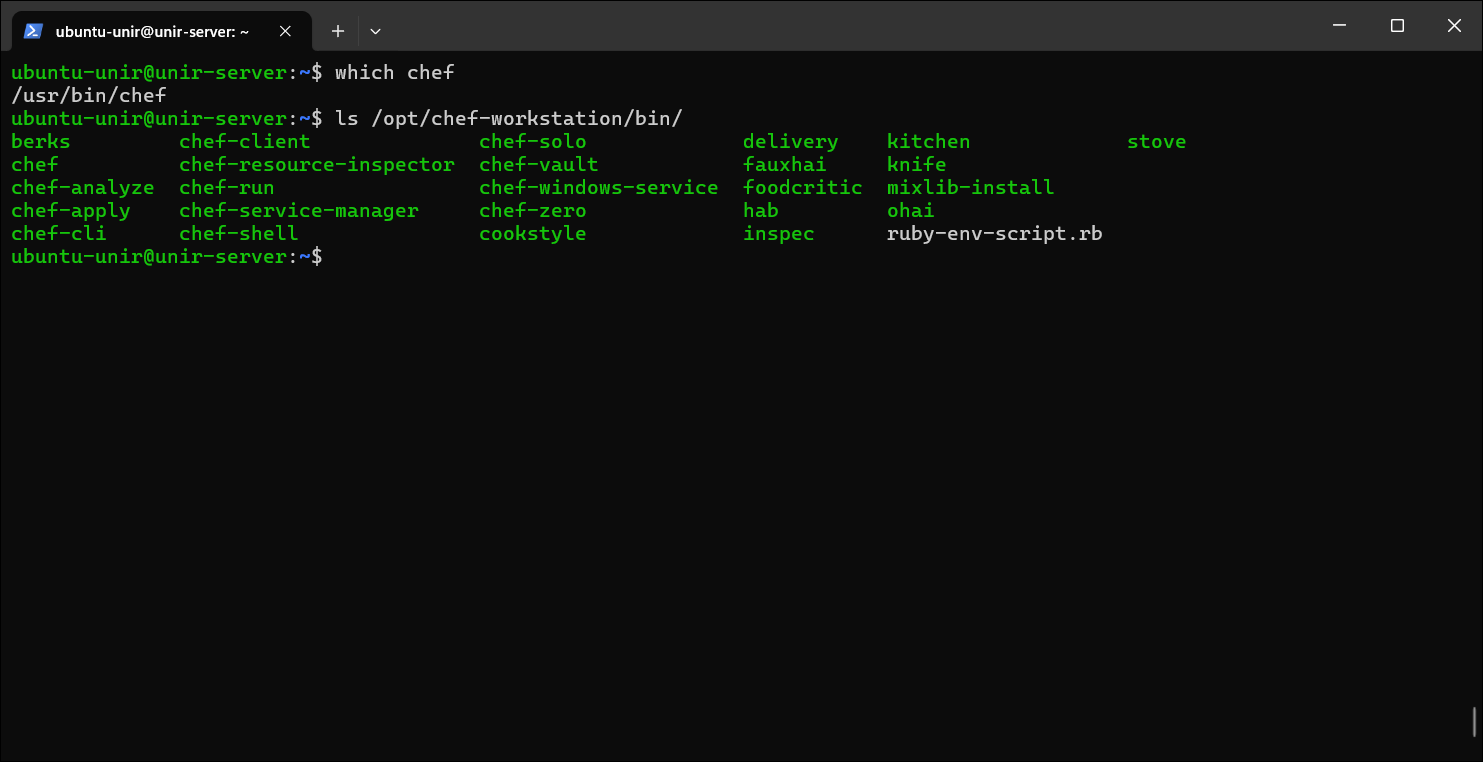
## Verificación del binario y estructura interna

Para asegurarse de que los ejecutables de Chef estaban correctamente ubicados, se ejecutaron los siguientes comandos:

which chef

ls /opt/chef-workstation/bin/

Este último comando listó todos los ejecutables incluidos con la instalación, como ***chef-client***, ***chef-run***, ***knife***, ***inspec***, ***kitchen***, entre otros.



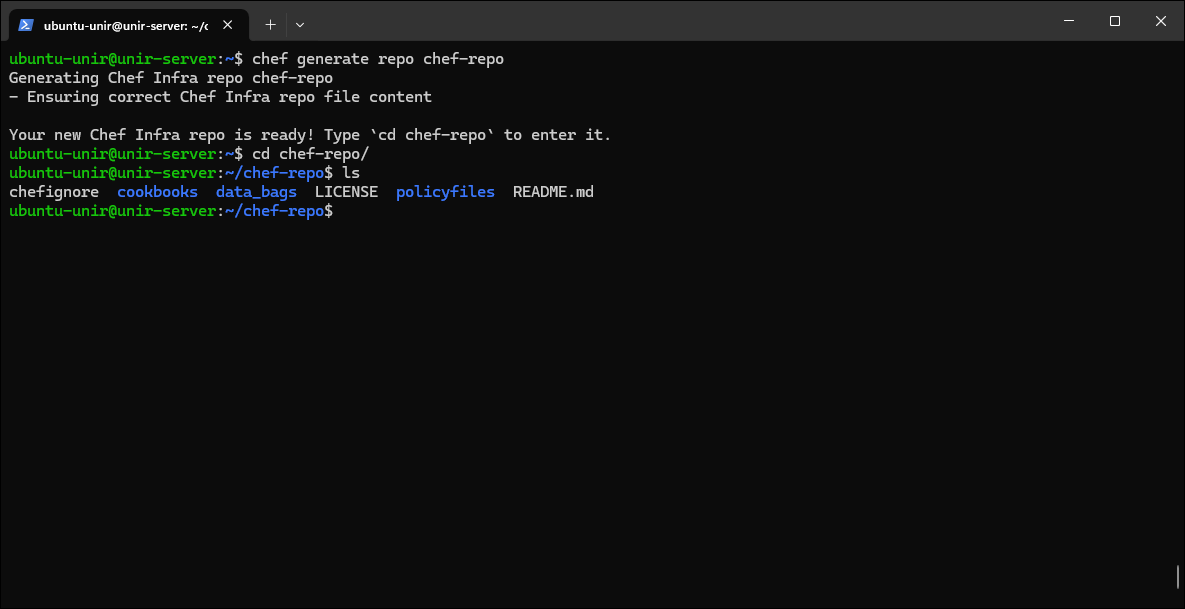
## Generación de estructura base para trabajo con recetas

Para asegurarse de que los ejecutables de Chef estaban correctamente ubicados, se ejecutaron los siguientes comandos:

chef generate repo chef-repo

cd chef-repo

Esto creó un directorio ***chef-repo*** con la estructura base que incluye carpetas para cookbooks, data bags y otras entidades propias del entorno Chef.



La instalación y configuración de Chef Workstation se realizó de manera exitosa. Todos los binarios fueron correctamente reconocidos, y se estableció el entorno base de trabajo para la posterior creación de recetas y cookbooks. Esta configuración servirá como punto de partida para las siguientes fases de la actividad.

# Cookbooks

## Cookbook 1: fastapi\_app

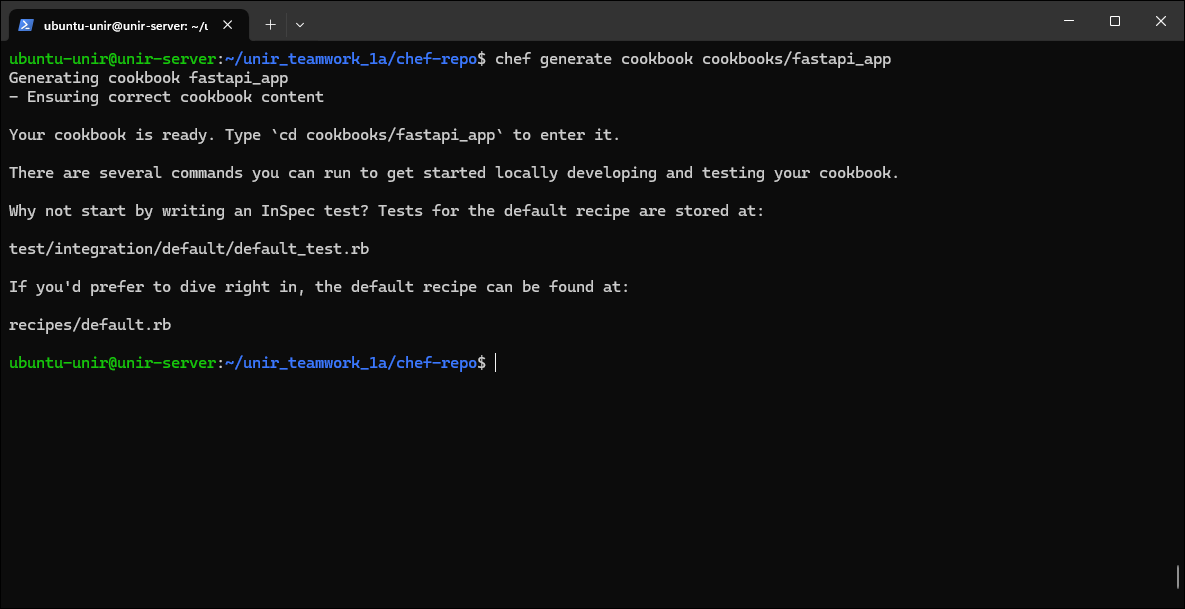
Cookbook personalizado llamado fastapi\_app con el objetivo de automatizar el despliegue de una aplicación web FastAPI sobre un servidor Ubuntu. El proceso incluyó:

* Instalación de herramientas necesarias.
* Creación de un entorno virtual de Python.
* Instalación de dependencias.
* Generación del código fuente (main.py) mediante plantilla.
* Ejecución automatizada de la aplicación mediante un servicio systemd.

### Paso 1: Crear el cookbook

Desde la terminal, en el directorio ***chef-repo***, se ejecutó el siguiente comando:

chef generate cookbook cookbooks/fastapi\_app



Este comando generó la estructura básica del cookbook ***fastapi\_app***.



### Paso 2: Crear el archivo de plantilla main.py

Se creó el directorio de plantillas y el archivo:

mkdir -p cookbooks/fastapi\_app/templates/default

nano cookbooks/fastapi\_app/templates/default/main.py.erb

Con el siguiente contenido:

from fastapi import FastAPI

app = FastAPI(

*title*="Demo FastAPI for Chef Workstation",

*description*="API de demostración para la tarea del grupo 1\_A",

*version*="1.0.0"

)

@app.get("/")

*def* read\_root():

    return {

        "message": "Hello from FastAPI"

    }

@app.get("/healthcheck")

*def* healthcheck():

    return {

        "status": "ok"

    }

@app.get("/about")

*def* about():

    return {

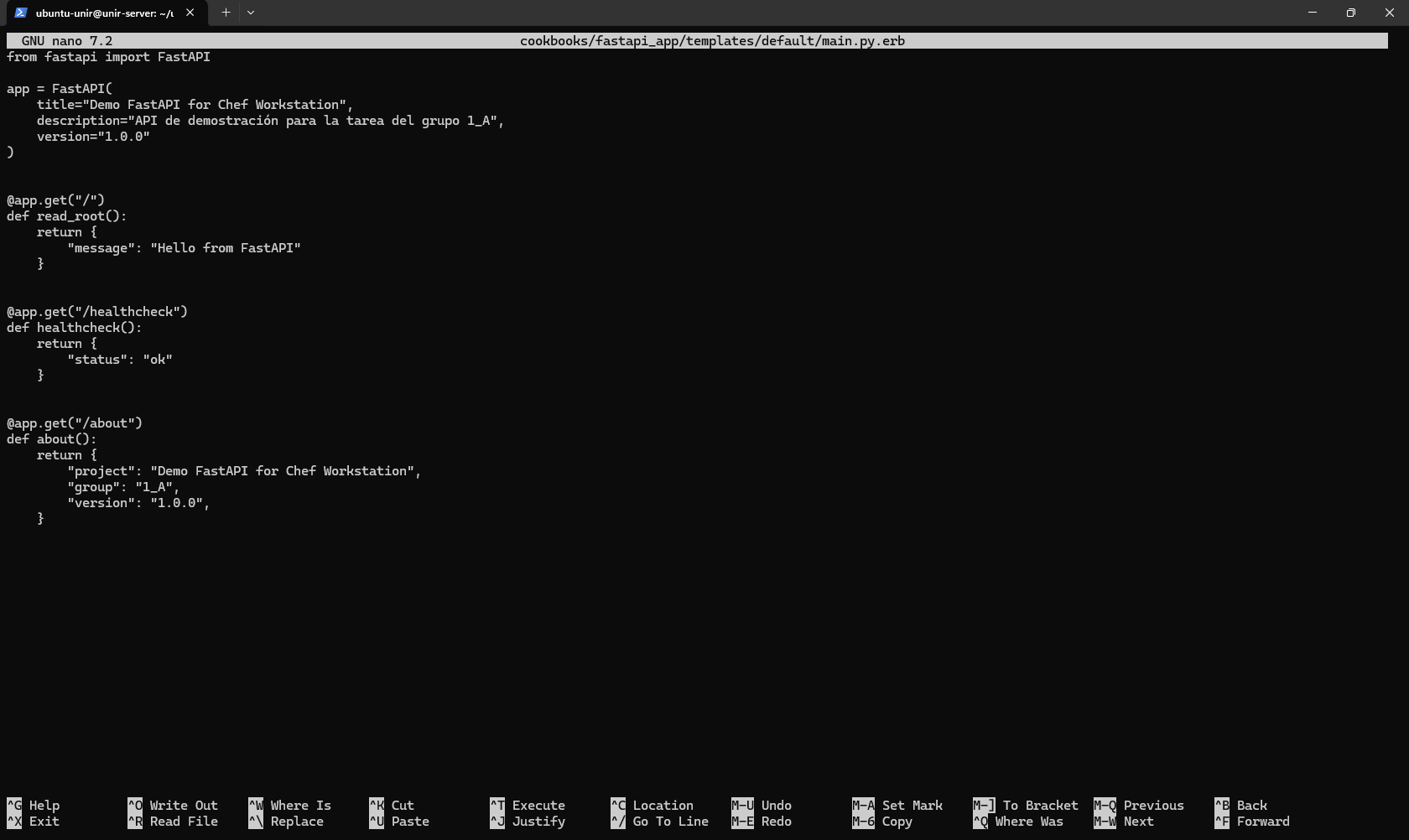
        "project": "Demo FastAPI for Chef Workstation",

        "group": "1\_A",

        "version": "1.0.0",

    }

Con el siguiente contenido:



### Paso 3: Editar la receta default.rb

Con el cookbook ***fastapi\_app*** ya generado y la plantilla ***main.py.erb*** lista, se procedió a construir la receta principal ***default.rb***, ubicada en:

cookbooks/fastapi\_app/recipes/default.rb

Esta receta fue organizada en bloques funcionales claramente definidos, utilizando recursos nativos de Chef para asegurar una implementación idempotente, ordenada y reutilizable.

**Actualizar el sistema y preparar el entorno**

Se actualiza el sistema operativo para evitar conflictos con versiones obsoletas de paquetes y se instalan las herramientas necesarias:

execute 'apt\_update' do

  command 'apt update && apt upgrade -y'

end

package %w(python3-venv curl) do

  action :install

end

**Crear el directorio del proyecto**

directory '/home/ubuntu-unir/fastapi\_app' do

  owner 'ubuntu-unir'

  group 'ubuntu-unir'

  mode '0755'

  recursive true

  action :create

end

Este directorio contendrá todo el proyecto: entorno virtual, archivo de aplicación y scripts.

**Crear entorno virtual**

execute 'crear entorno virtual' do

  command 'python3 -m venv .venv'

  cwd '/home/ubuntu-unir/fastapi\_app'

  user 'ubuntu-unir'

  environment({ 'HOME' => '/home/ubuntu-unir' })

  not\_if {

::File.exist?('/home/ubuntu-unir/fastapi\_app/.venv/bin/activate')

}

end

Se crea el entorno solo si no existe, y se ejecuta como el usuario ***ubuntu-unir*** para evitar errores de permisos.

**Instalar FastAPI**

execute 'instalar fastapi' do

  command '/home/ubuntu-unir/fastapi\_app/.venv/bin/pip install "fastapi[standard]"'

  cwd '/home/ubuntu-unir/fastapi\_app'

  user 'ubuntu-unir'

  environment({

    'HOME' => '/home/ubuntu-unir',

    'PATH' => '/home/ubuntu-unir/fastapi\_app/.venv/bin:/usr/bin:/bin'

  })

end

Se instalan las dependencias en el entorno virtual, asegurando aislamiento del sistema principal

**Generar main.py desde plantilla**

template '/home/ubuntu-unir/fastapi\_app/main.py' do

  source 'main.py.erb'

  owner 'ubuntu-unir'

  group 'ubuntu-unir'

  mode '0644'

end

Este archivo define los endpoints ***/***, ***/healthcheck*** y ***/about*** de la aplicación FastAPI.

**Crear servicio systemd fastapi.service**

file '/etc/systemd/system/fastapi.service' do

  content <<~UNIT

    [Unit]

    Description=FastAPI Application

    After=network.target

    [Service]

    User=ubuntu-unir

    WorkingDirectory=/home/ubuntu-unir/fastapi\_app

    ExecStart=/home/ubuntu-unir/fastapi\_app/.venv/bin/python -m fastapi run main.py --host 0.0.0.0 --port 8080

    Restart=always

    [Install]

    WantedBy=multi-user.target

  UNIT

  mode '0644'

end

Este archivo define cómo se ejecutará la aplicación como servicio del sistema, de forma persistente y con reinicio automático en caso de falla.

**Recargar systemd y activar el servicio**

execute 'reload systemd' do

  command 'systemctl daemon-reexec && systemctl daemon-reload'

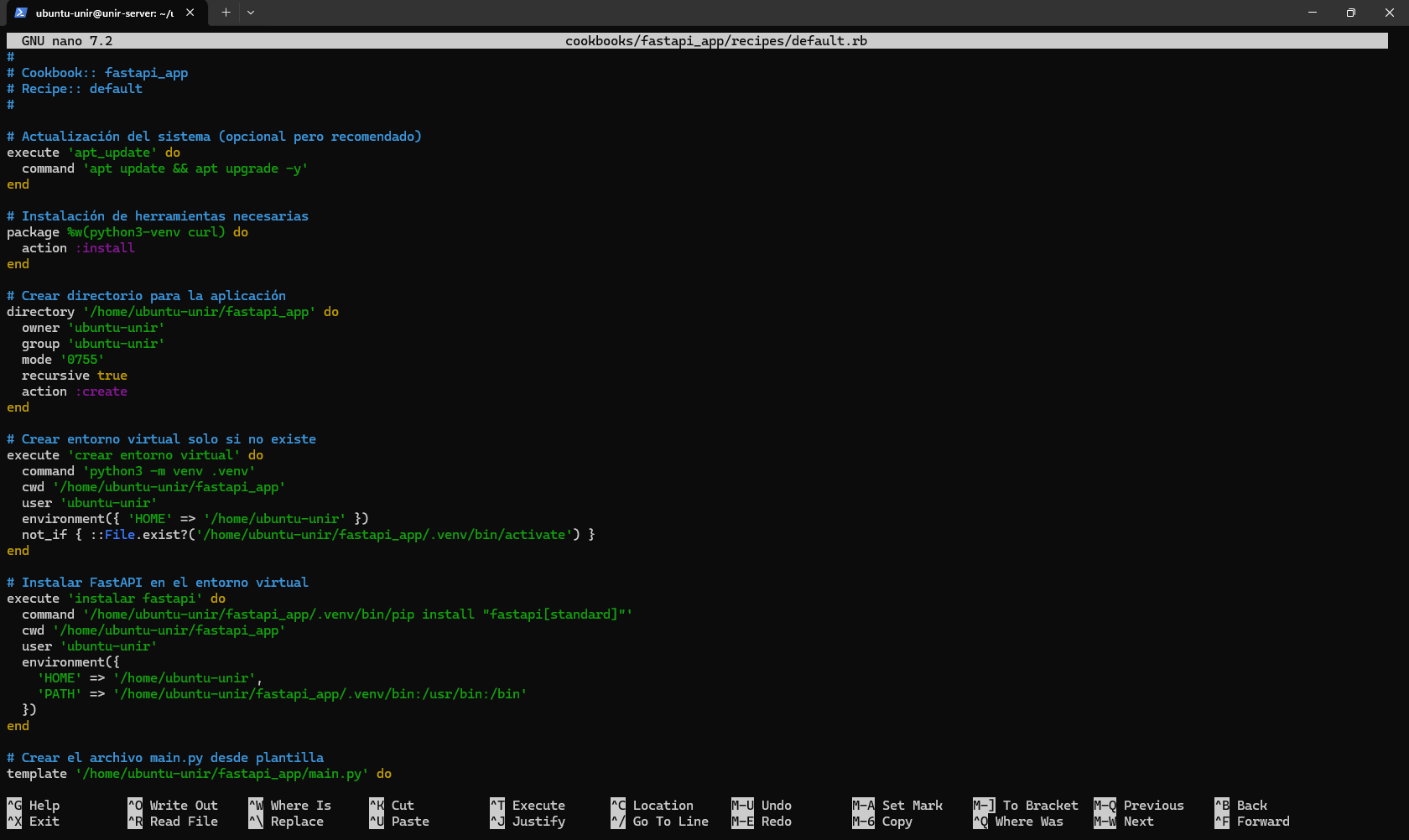
end

service 'fastapi' do

  action [:enable, :start]

end

Con esto, el servicio ***fastapi*** queda habilitado para iniciarse automáticamente con el sistema y se activa de inmediato.



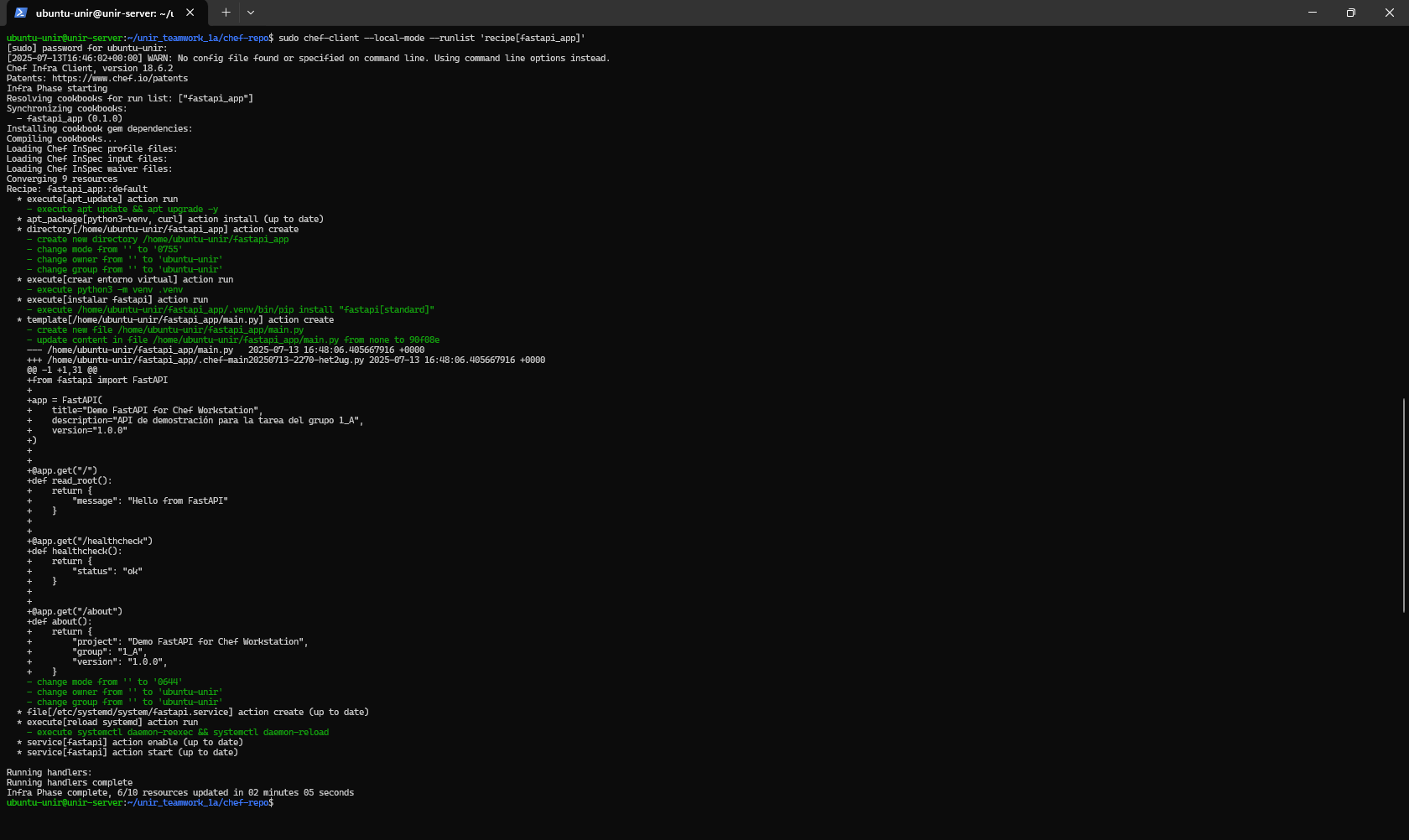
### Paso 4: Ejecución del cookbook y validación del despliegue

Una vez completada la receta ***default.rb***, se procedió a su ejecución para desplegar la aplicación FastAPI de forma automatizada.

**Ejecutar el cookbook con Chef**

sudo chef-client --local-mode --runlist 'recipe[fastapi\_app]'

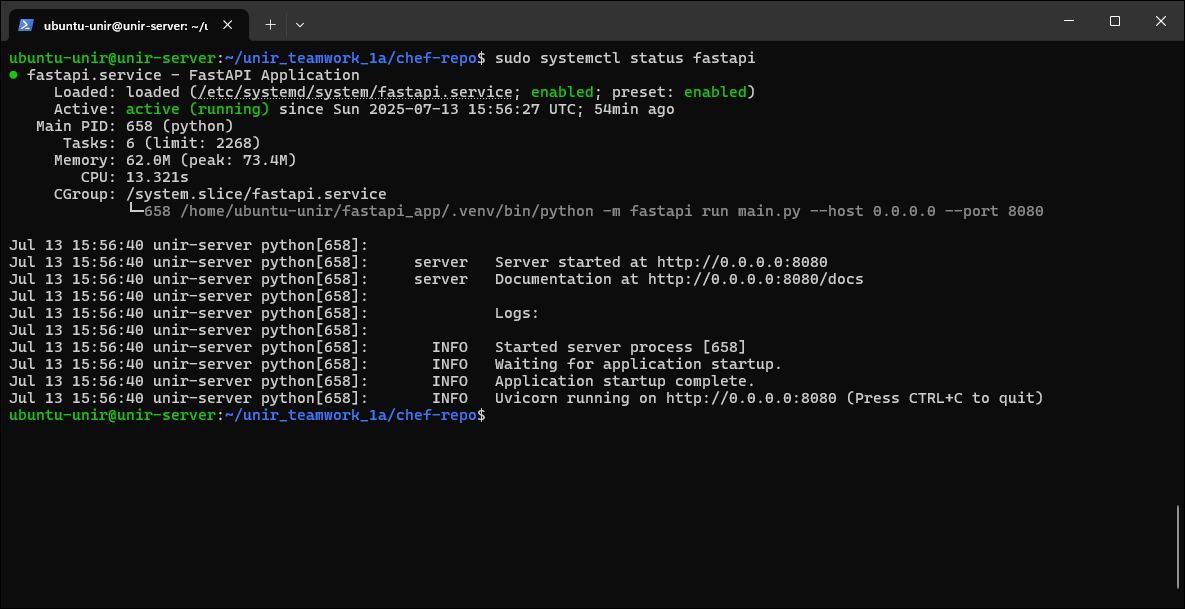
Este comando indica a Chef que ejecute en modo local (***--local-mode***) y aplique específicamente la receta fastapi\_app.



**Verificar que el servicio esté activo**

Después de aplicar la receta, se validó que el servicio fastapi estuviera activo con:

sudo systemctl status fastapi



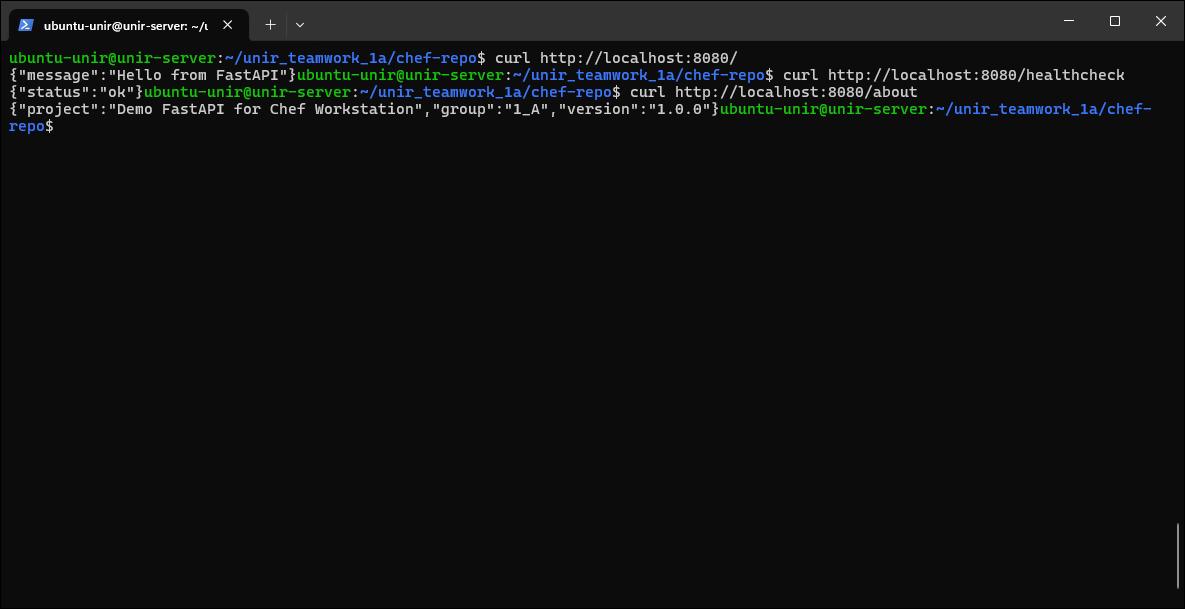
**Validar acceso a la API**

Se realizaron peticiones HTTP locales para confirmar que la aplicación FastAPI se encontraba en funcionamiento y respondiendo correctamente a sus rutas definidas.

curl http://localhost:8080/

curl http://localhost:8080/healthcheck

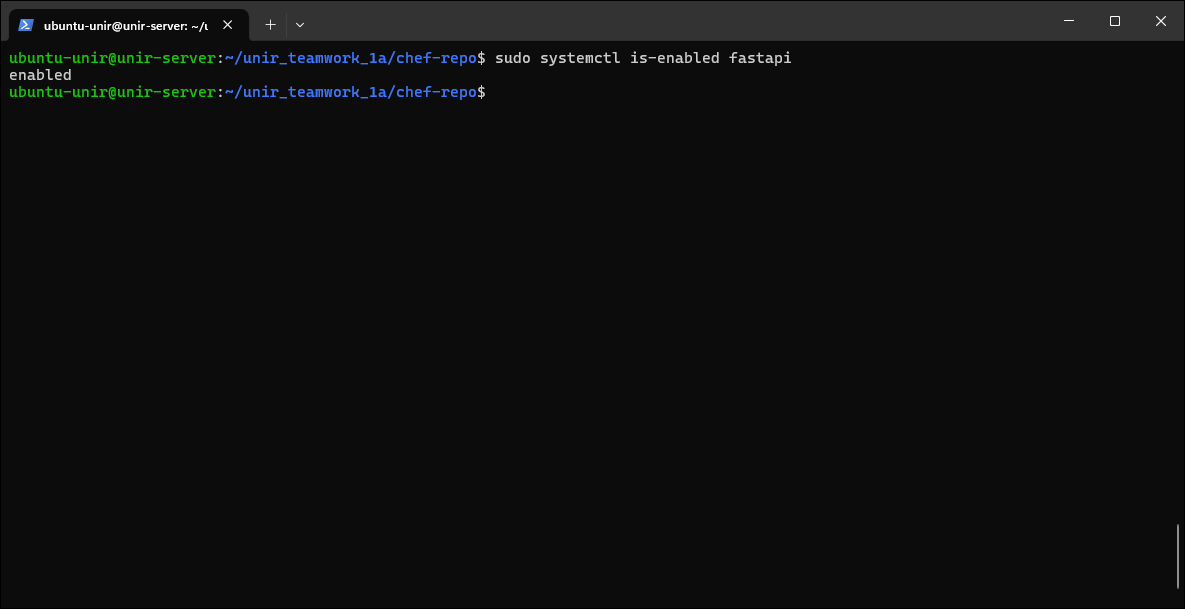
curl http://localhost:8080/about



**Verificar inicio automático del servicio**

Para confirmar que el servicio se inicia automáticamente con el sistema, se ejecutó:

sudo systemctl is-enabled fastapi



El despliegue fue exitoso. La aplicación FastAPI quedó:

* Ejecutándose en el puerto 8080.
* Expuesta como servicio systemd.
* Respondiendo a todas las rutas esperadas.
* Preparada para iniciar automáticamente al reiniciar el servidor.

Esto demuestra que la receta es funcional, reproducible y adecuada para producción o entornos educativos.

## Cookbook 2: Instalación de Ngrok

Cookbook personalizado llamado chef\_ngrok\_demo, cuyo objetivo fue automatizar la instalación y configuración de la herramienta Ngrok en un sistema macOS ARM64. El proceso incluyó:

* Descarga del binario ARM64 de Ngrok.
* Creación del archivo de configuración ngrok.yml con token de autenticación.
* Generación de un script de lanzamiento para abrir túneles hacia puertos locales.
* Validación del entorno mediante pruebas automatizadas con InSpec.

Ngrok es ampliamente utilizado en desarrollo web, pruebas de APIs, webhooks y entornos DevOps, ya que permite exponer servicios locales a Internet de forma rápida y segura sin necesidad de redireccionar puertos.

### Paso 1: Crear el cookbook

Desde la terminal, se creó el cookbook con el siguiente comando:

cd chef-repo/cookbooks

chef generate cookbook chef\_ngrok\_demo

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

### Paso 2: Editar la receta default.rb

Se editó el archivo ***recipes/default.rb*** para automatizar los siguientes procesos:

* Descargar el binario de Ngrok compatible con la arquitectura ARM64.
* Crear el archivo de configuración .ngrok2/ngrok.yml con token y región definidos.
* Generar un script llamado lanzar\_ngrok.sh que ejecuta el túnel a un puerto local.

### Paso 3: Ejecutar el cookbook

Desde la raíz del repositorio Chef se ejecutó el cookbook utilizando:

sudo chef-client --local-mode --runlist 'recipe[chef\_ngrok\_demo]'

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

### Paso 4: Lanzar el servidor local y script de Ngrok

Se lanzó un servidor HTTP local en el puerto 3000 para simular una aplicación:

cd ~

python3 -m http.server 3000

Posteriormente, se ejecutó el script generado para abrir el túnel:

./ngrok/lanzar\_ngrok.sh

### Pruebas automatizadas con InSpec

Para validar la instalación y configuración, se desarrollaron pruebas automatizadas en el archivo:

test/integration/default/default\_test.rb

**Código de prueba:**

describe file("#{ENV['HOME']}/ngrok/ngrok") do

it { should exist }

it { should be\_executable }

end

describe file("#{ENV['HOME']}/.ngrok2/ngrok.yml") do

it { should exist }

its('content') { should match /authtoken: dummy\_token\_para\_clase/ }

end

describe file("#{ENV['HOME']}/ngrok/lanzar\_ngrok.sh") do

it { should exist }

it { should be\_executable }

end

describe port(4040) do

it { should\_not be\_listening }

end

Estas pruebas garantizan que:

* El binario fue descargado correctamente.
* El archivo de configuración existe y contiene el token esperado.
* El script de ejecución es funcional.
* El puerto 4040 (dashboard web) no está escuchando por defecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

Ejecución:

chef exec inspec exec test/integration/default

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

## Cookbook 3: Instalación y despliegue de Jenkins automatizado

Cookbook personalizado llamado jenkins, cuyo objetivo fue automatizar la instalación y despliegue de un servidor Jenkins en un contenedor Docker. El proceso se diseñó para ejecutarse sobre sistemas Linux compatibles (Debian o RHEL), donde la instalación de Docker es sencilla desde terminal.

Este cookbook permite preparar rápidamente un entorno de integración continua con Jenkins, obteniendo la contraseña inicial del administrador automáticamente desde Chef y dejando el servicio listo para configuración. El proceso incluyó:

* Instalación automática de Docker (si no está instalado).
* Eliminación de cualquier contenedor Jenkins existente.
* Despliegue de una nueva instancia Jenkins LTS en Docker.
* Espera activa hasta que el contenedor genere la contraseña inicial.
* Lectura y despliegue del valor de initialAdminPassword desde Chef.
* Validación del estado de Jenkins accediendo a su interfaz web.

### Paso 1: Crear el cookbook

Se generó el cookbook con el comando:

chef generate cookbook cookbooks/jenkins

### Paso 2: Editar la receta default.rb

Se creó el archivo ***recipes/default.rb*** e incorporaron los siguientes bloques clave:

**Definición de variables:** Se define una variable reutilizable para el puerto de exposición de Jenkins:

#variables reutilizables

jenkins\_port = '8080'

**Detección del sistema operativo:** Para asegurar compatibilidad con Linux, se imprime el sistema operativo detectado:

#detectar sistema operativo y mostrarlo

ruby\_block 'print\_os' do

block do

puts "\n viendo sistema operativo\n"

platform = node['platform']

puts "\n sistema operativo detectado: #{platform}\n"

end

end

**Instalación automática de Docker (si no está instalado):** Chef incluye una extensión de docker, pero por la compatibilidad y versión actual de Jenkins ésta no podía ser utilizada, así que mejor se optó por usar la opción de Heredocs para su instalación.

#instalar docker si no está instalado( Linux)

if node['platform\_family'] == 'debian' || node['platform\_family'] == 'rhel'

execute 'install\_docker' do

puts "\n instalando docker \n"

command <<-EOH

curl -fsSL <https://get.docker.com> | sh

EOH

not\_if 'which docker'

end

end

**Eliminación de Jenkins existente (si aplica):** En este caso para propósito de pruebas se generó un bloque donde se revisa si el servicio de jenkins se encontraba ya instalado y corriendo, y de ser así lo pausara y borrara. Esto permite que la receta sea reutilizable sin errores por contenedores previos.

#detectar si jenkins ya se está ejecutando y detener

execute 'stop\_and\_remove\_jenkins' do

command <<-EOH

sudo docker stop jenkins || true

sudo docker rm jenkins || true

EOH

only\_if "docker inspect jenkins >/dev/null 2>&1"

end

Despliegue de Jenkins LTS en Docker

#Ejecutar jenkins en Docker

execute 'run\_jenkins\_container' do

command <<-EOH

puts "ejecutando docker-jenkins"

docker run -d --name jenkins \

-p #{jenkins\_port}:8080 \

jenkins/jenkins:lts

EOH

not\_if 'docker ps | grep jenkins'

end

**Espera activa hasta que Jenkins esté listo**: Este bloque evita errores por intentar acceder a Jenkins antes de estar inicializado.

execute 'wait\_for\_jenkins\_container' do

command <<-EOH

while ! docker exec jenkins test -f /var/jenkins\_home/secrets/initialAdminPassword; do

sleep 2

done

EOH

retries 10

retry\_delay 3

end

Mostrar contraseña inicial del administrador

ruby\_block 'print\_initial\_admin\_password' do

block do

password = %x(sudo docker exec jenkins cat /var/jenkins\_home/secrets/initialAdminPassword).strip

puts "\n Jenkins initial admin password: #{password}\n"

end

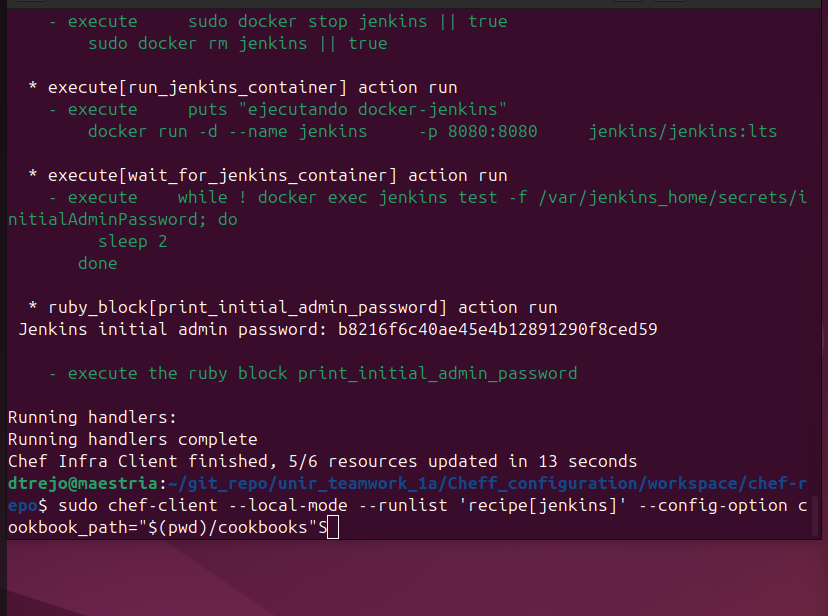
end

### Paso 3: Ejecutar la receta

La receta se ejecuta desde la raíz del repositorio con:

sudo chef-client --local-mode --runlist 'recipe[jenkins]' --config-options cookbook\_path="${PWD}/cookbooks"

Como primera prueba podemos observar que en la salida de terminal la receta nos da la contraseña para el primer inicio de Jenkins.

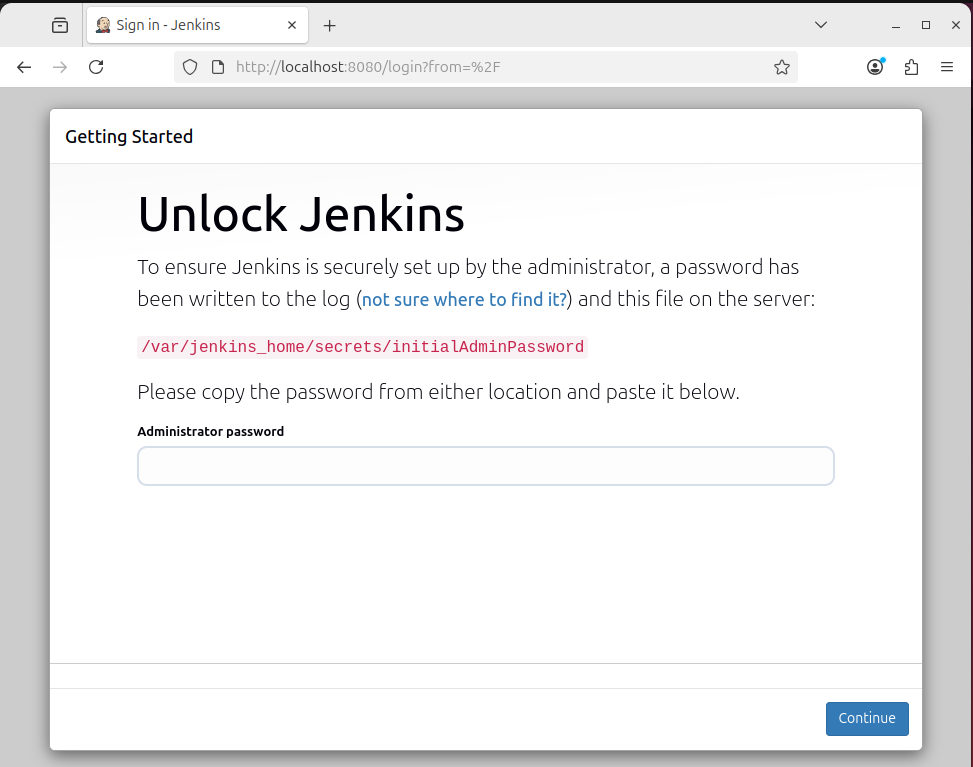


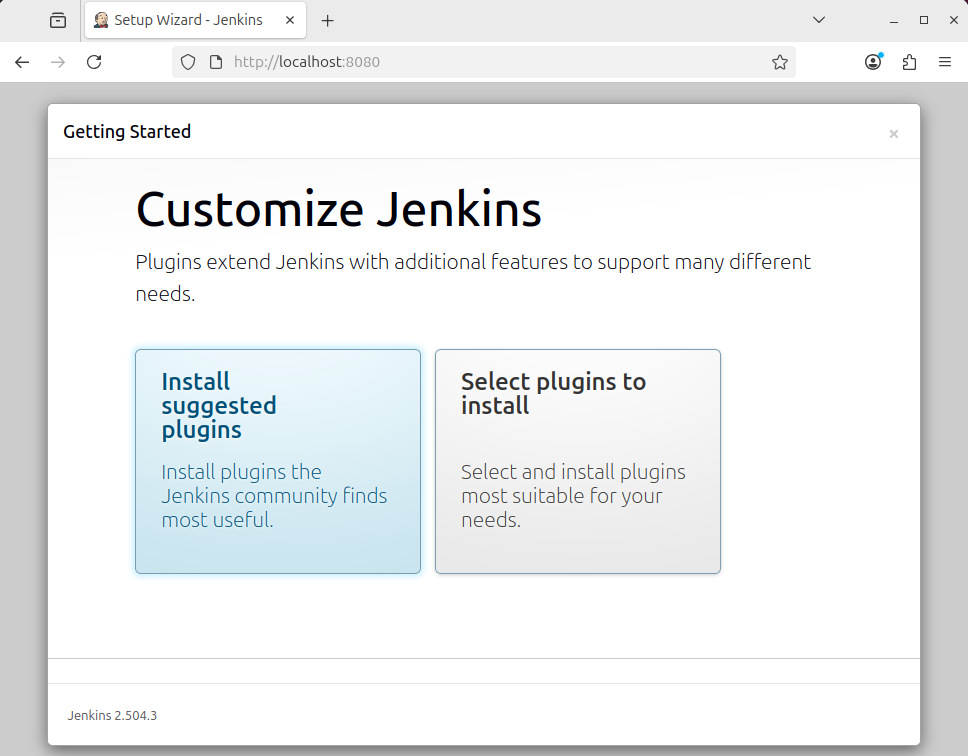
### Paso 4: Validar Jenkins en el navegador

Para poder verificar la salida del Jenkins Master listo para ser configurado tenemos que abrir el explorador y localizar el localhost con el puerto que elegimos

<http://localhost:8080>

Como segunda comprobación podemos observar que se esté mostrando la página inicial de Jenkins





Para completar la configuración inicial, se utiliza la contraseña impresa por Chef (initialAdminPassword), pegándola en el formulario de configuración del navegador.

**Reusabilidad de la receta:** Si se vuelve a ejecutar la receta, se elimina el contenedor anterior y se crea uno nuevo. Esto devuelve Jenkins al estado inicial, solicitando nuevamente la contraseña al acceder vía navegador.

**Resultado final esperado:**

* Docker instalado automáticamente si no estaba presente.
* Jenkins desplegado en contenedor Docker.
* Contraseña inicial disponible directamente en consola Chef.
* Jenkins accesible por navegador desde el puerto 8080.
* Capacidad de reiniciar Jenkins eliminando el contenedor anterior desde Chef.

## Cookbook 4: WordPress

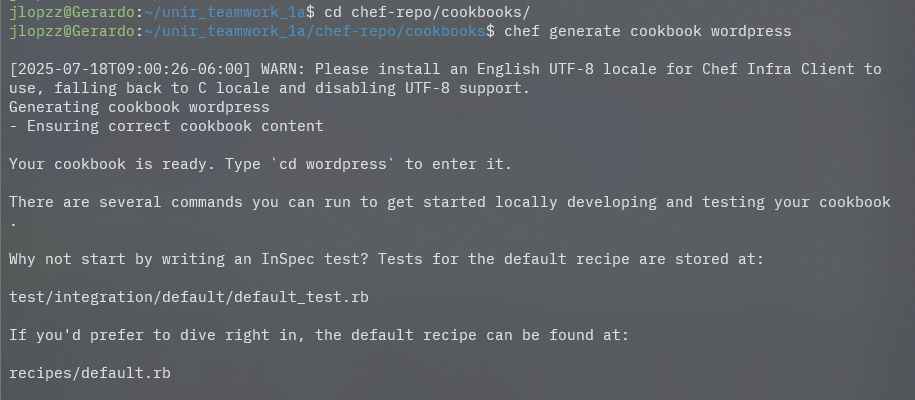
Cookbook personalizado llamado ***wordpress***, cuyo objetivo fue automatizar el despliegue de una instancia básica de WordPress sobre un servidor Linux con Apache, PHP y MySQL. El enfoque principal fue asegurar que todas las dependencias estén instaladas, configuradas correctamente y que WordPress sea accesible desde el navegador al finalizar la ejecución.

Este ejemplo demuestra cómo utilizar Chef para aprovisionar un stack web clásico (LAMP) de forma declarativa, asegurando que los servicios estén activos y que los archivos de WordPress estén en la ubicación esperada con los permisos adecuados.

Para este ejemplo se despliega una aplicación inicial de Wordpress, verificando si en el servidor existen las dependencias necesarias. De no existir, las instala.

### Paso 1: Crear el cookbook

Desde el directorio de cookbooks, se ejecutó el siguiente comando:



### Paso 2: Editar la receta default.rb

Se editó el archivo ***recipes/default.rb*** desde un entorno de desarrollo como Visual Studio Code. La receta contiene los siguientes bloques de automatización:



**Instalación de Apache y activación del servicio:** Esto garantiza que el servidor web esté instalado y activo:

*# Instalar Apache*

package *'apache2'* do

action :install

end

service *'apache2'* do

action [:enable, :start]

end

**Instalación de PHP y extensiones necesarias**: Se define un arreglo con las dependencias necesarias para que el servidor web ejecute código PHP y se conecte a una base de datos MySQL. Dicho arreglo se recorre con ***each***:

*# Instalar PHP y extensiones necesarias*

*%w(php libapache2-mod-php php-mysql)*.each do |pkg|

package pkg do

action :install

end

end

Instalación de MySQL y activación del servicio:

*# Instalar MySQL Server*

package *'mysql-server'* do

action :install

end

service *'mysql'* do

action [:enable, :start]

end

Creación de base de datos y usuario de WordPress

*# Crear base de datos y usuario para WordPress*

execute *'create-wordpress-user-and-db'* do

command *<<-EOH*

*mysql -e "CREATE DATABASE IF NOT EXISTS wordpress;"*

*mysql -e "CREATE USER IF NOT EXISTS 'wp\_user'@'localhost' IDENTIFIED BY 'wp\_pass';"*

*mysql -e "GRANT ALL PRIVILEGES ON wordpress.\* TO 'wp\_user'@'localhost';"*

*mysql -e "FLUSH PRIVILEGES;"*

*EOH*

not\_if *"mysql -e \"SHOW DATABASES LIKE 'wordpress';\" | grep wordpress"*

end

Descarga del paquete de WordPress:

*# Descargar WordPress*

remote\_file *'/tmp/wordpress.tar.gz'* do

source *'*[*https://wordpress.org/latest.tar.gz*](https://wordpress.org/latest.tar.gz)*'*

action :create

end

**Extracción, configuración y permisos:** Se desempaqueta el archivo y se copia a /var/www/html/, eliminando el archivo de inicio de Apache por defecto:

*# Extraer y mover WordPress*

bash *'extract\_wordpress'* do

cwd *'/tmp'*

code *<<-EOH*

*tar -xzf wordpress.tar.gz*

*cp -r wordpress/\* /var/www/html/*

*chown -R www-data:www-data /var/www/html/*

*chmod -R 755 /var/www/html/*

*rm /var/www/html/index.html*

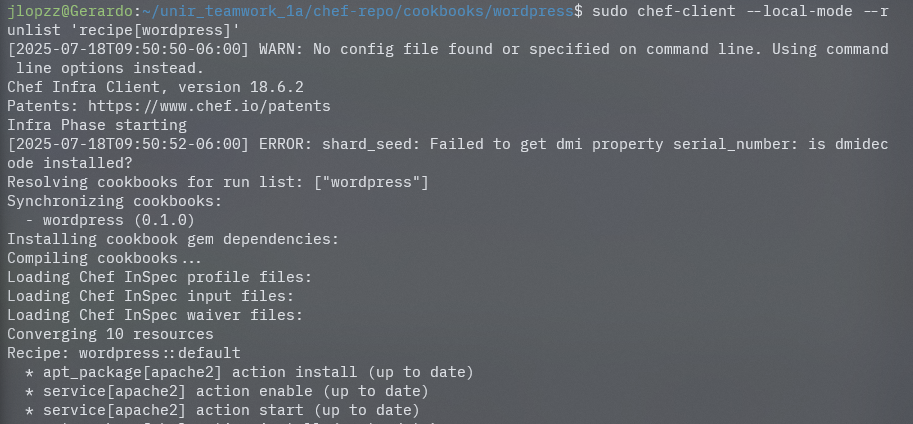
*EOH*

not\_if { ::File.exist?(*'/var/www/html/wp-config.php'*) }

end

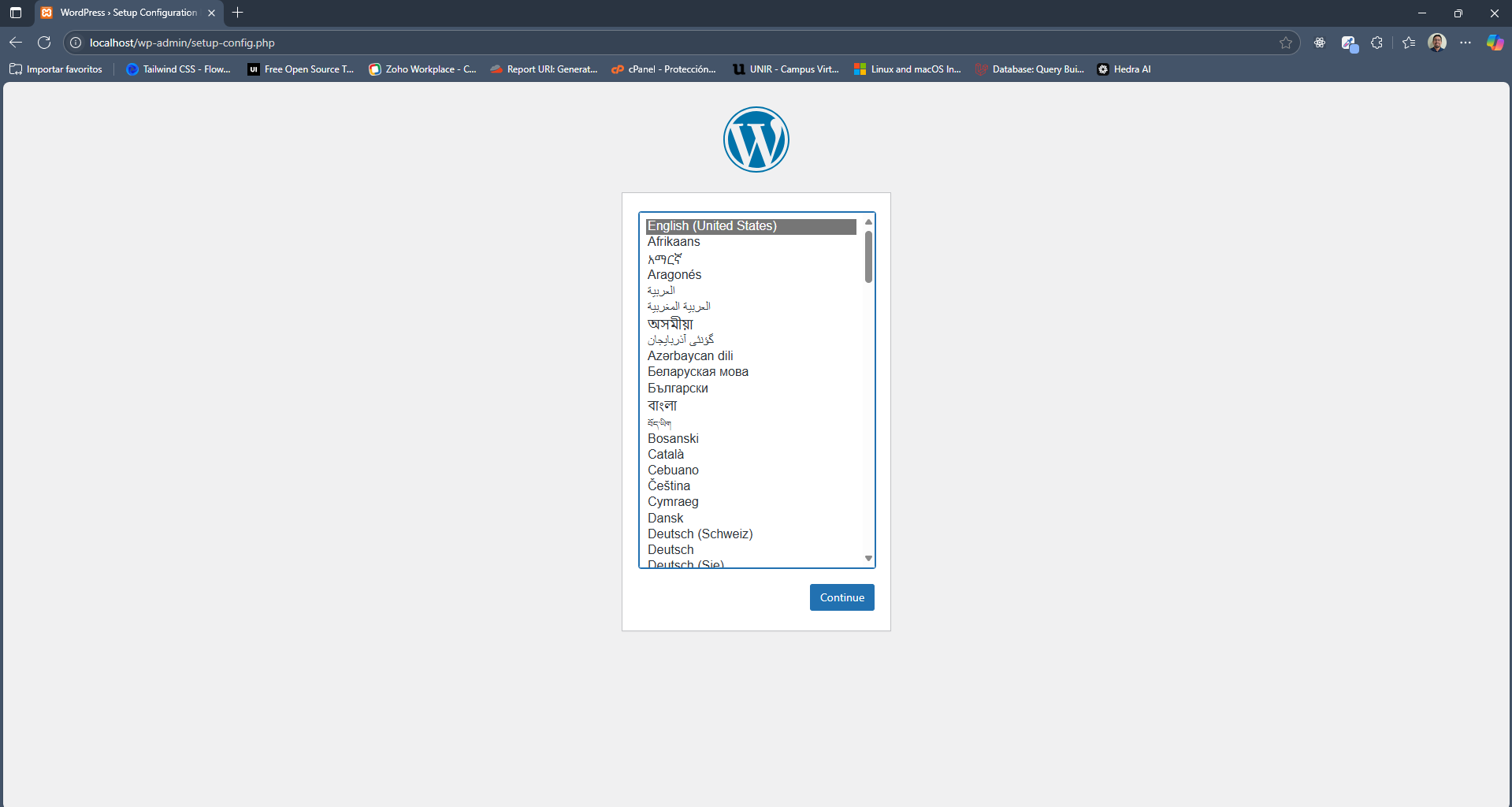
### Paso 3: Ejecutar el cookbook

Una vez finalizada la receta, se ejecutó el cliente Chef con permisos de administrador:

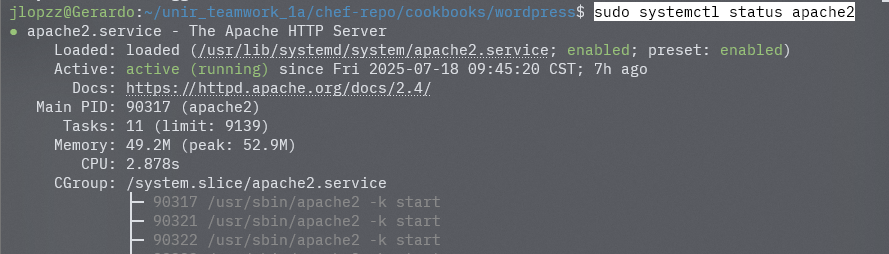


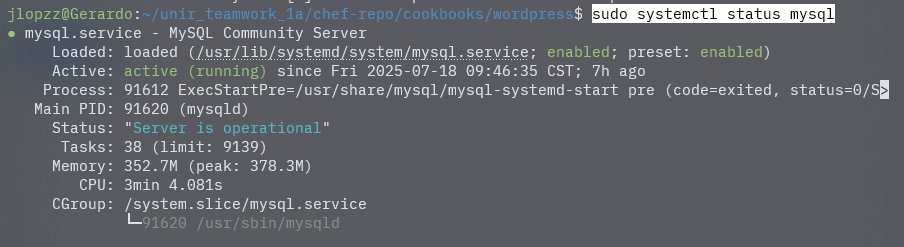
### Paso 4: Validación del despliegue

Para finalizar, ya se han instalado todas y cada una de las distintas dependencias, solo se ingresa a la dirección o dominio del servidor en donde estaría corriendo nuestro WordPress, y lo veremos ejecutándose de manera exitosa:



También se puede verificar el estado de los servicios requeridos (Apache y MySQL) con:





## Cookbook 5: RunSpringbootAPP

Cookbook personalizado llamado RunSpringbootAPP, cuyo objetivo fue automatizar el despliegue de una aplicación web Java Spring Boot desde un repositorio remoto en un entorno macOS. Este ejemplo demuestra cómo configurar una receta Chef para clonar, construir y ejecutar una aplicación basada en Java, de forma automática, asegurando su disponibilidad desde el navegador.

**El proceso incluye:**

* Definición de variables importantes.
* Instalación de dependencias usando brew.
* Clonado de un repositorio público.
* Construcción del JAR.
* Eliminar un proceso ya existente de java en caso de que lo haya.
* Ejecución automatizada de la aplicación mediante Java.

### Paso 1: Crear el cookbook

chef generate cookbook cookbooks/RunSpringbootAppCookbook 3

### Paso 2: Seleccionar el repositorio de ejemplo

Para este ejercicio se utilizó el repositorio oficial de Spring Boot disponible en:

<https://github.com/spring-guides/gs-spring-boot/tree/main>

Este proyecto inicia un servidor web en el puerto 8080. Al acceder a la raíz del servidor (GET /), responde con el mensaje: "Greetings from Spring Boot!".

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

El contenido generado por IA puede ser incorrecto.

### Paso 3: Editar la receta default.rb

Dentro del archivo ***recipes/default.rb*** se implementaron los siguientes pasos, comentados y ordenados por propósito:

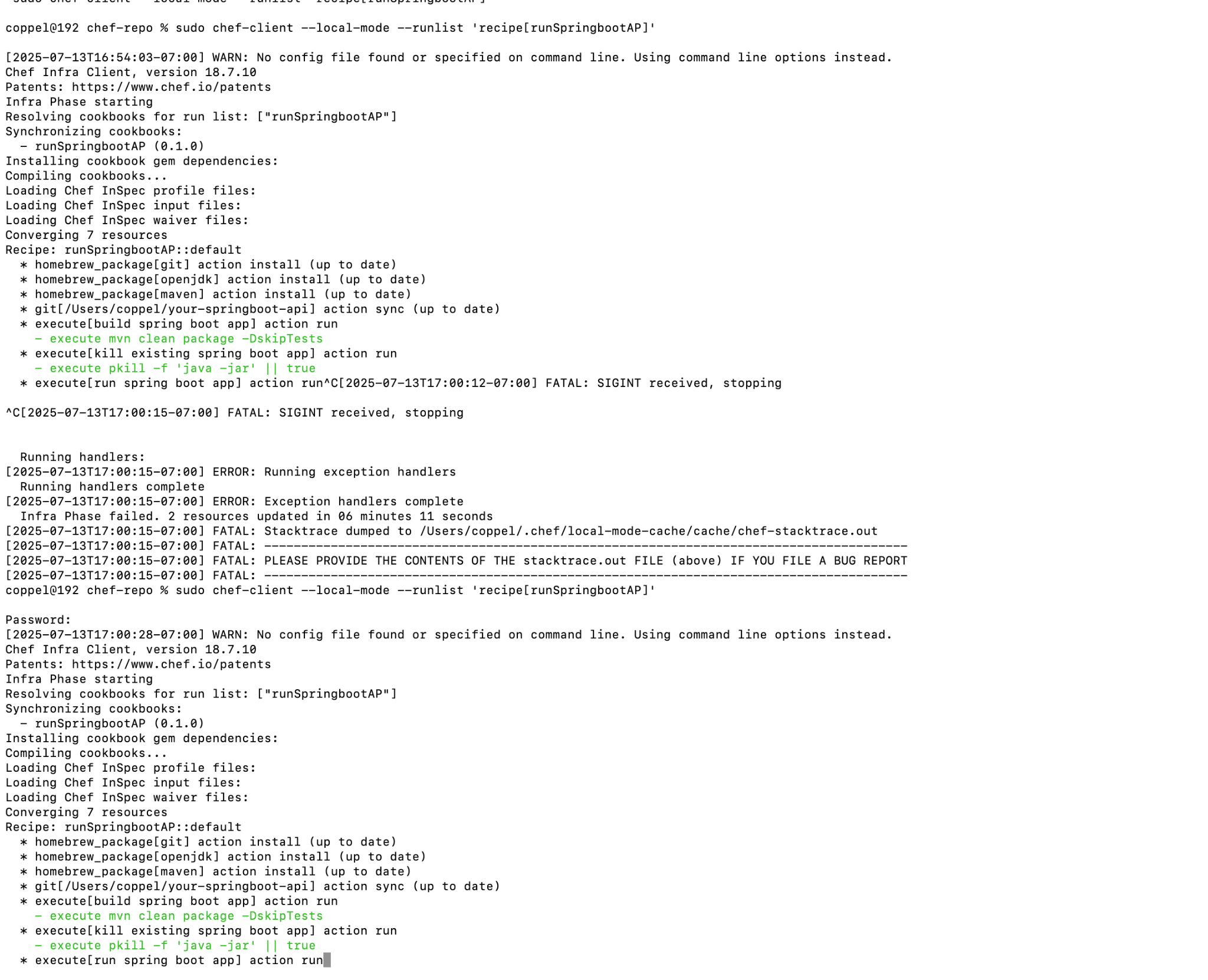
**Texto

El contenido generado por IA puede ser incorrecto.**

### Paso 4: Ejecutar el cookbook

Una vez completada la receta, se ejecutó con:

sudo chef-client --local-mode --runlist 'recipe[RunSpringbootAPP]'

****

### **Paso 5: Validación en el navegador**

Con la aplicación ejecutándose, se accedió al navegador con la dirección:

http://localhost:8080

Y se validó la respuesta esperada:

Greetings from Spring Boot!

Interfaz de usuario gráfica, Texto, Aplicación

El contenido generado por IA puede ser incorrecto.

**Resultado final esperado**

* Java instalado correctamente con Homebrew.
* Repositorio Spring Boot clonado localmente.
* Aplicación construida exitosamente con Maven Wrapper.
* Servidor corriendo en el puerto 8080 de forma automatizada.
* Validación exitosa desde el navegador con la respuesta esperada.

# Repositorio con los cookbooks

<https://github.com/DavidTrF/unir_teamwork_1a>

# Tabla de valoración

|  |  |  |  |
| --- | --- | --- | --- |
|  | Sí | No | A veces |
| Todos los miembros se han integrado al trabajo del grupo | X |  |  |
| Todos los miembros participan activamente | X |  |  |
| Todos los miembros respetan otras ideas aportadas | X |  |  |
| Todos los miembros participan en la elaboración del informe | X |  |  |
| Me he preocupado por realizar un trabajo cooperativo con mis compañeros | X |  |  |
| Señala si consideras que algún aspecto del trabajo en grupo no ha sido adecuado | X |  |  |

# Reflexión sobre los desafíos enfrentados

Durante el desarrollo de la actividad, el equipo enfrentó varios desafíos técnicos y organizativos que requirieron investigación, comunicación y toma de decisiones colaborativa. A continuación, se enumeran los principales obstáculos encontrados:

* **Configuración inicial de Chef Workstation en Ubuntu Server:** Uno de los principales obstáculos fue la configuración correcta del entorno de Chef Workstation en Ubuntu Server, especialmente en lo relativo a la disponibilidad del comando chef, las rutas del binario y la necesidad de configurar correctamente el entorno de shell para evitar errores como command not found.
* **Problemas al intentar usar nohup desde el cookbook 1 (fastapi\_app):** Se descubrió que Chef no puede gestionar correctamente procesos que quedan en segundo plano con nohup, ya que este comando bloqueaba la ejecución de la receta. Se reemplazó por una solución más robusta con la creación de un servicio systemd, que permite iniciar, detener y monitorear la aplicación de forma segura.
* **Idempotencia de la receta Chef:** Algunos comandos se debieron condicionar con not\_if para evitar reejecuciones innecesarias y garantizar que la receta fuera idempotente (repetible sin efectos adversos).
* **Adaptación del cookbook 2 (chef\_ngrok\_demo) a macOS ARM64:** En entornos distintos a Ubuntu, se encontraron limitaciones específicas. En particular, algunos binarios como osquery no estaban disponibles para macOS con arquitectura ARM64. Esto obligó a replantear el objetivo inicial y buscar una alternativa compatible. Se optó por trabajar con Ngrok, lo cual demostró la importancia de saber adaptar soluciones a plataformas reales y con restricciones técnicas.
* **Validación automatizada con InSpec:** Para el cookbook de Ngrok, se exploró el uso de InSpec para realizar pruebas automáticas que verificaran la existencia de archivos, su contenido, permisos y puertos expuestos. Esto permitió comprobar que la configuración aplicada por la receta cumplía con lo esperado, reforzando así la calidad de la automatización y la confiabilidad del entorno.

# Conclusiones

* La herramienta Chef Workstation demostró ser poderosa y flexible para la automatización de despliegues, siempre que se estructure correctamente la lógica en cookbooks bien definidos.
* Es posible automatizar no solo la instalación de paquetes, sino también el despliegue completo de aplicaciones modernas, incluyendo su ejecución como servicios del sistema.
* El enfoque modular y declarativo de Chef favorece la reproducibilidad, una característica clave para entornos DevOps, pruebas e infraestructura como código.
* La experiencia permitió al equipo desarrollar habilidades técnicas prácticas, trabajar en equipo y documentar un proceso técnico completo.

# Recomendaciones

* Para futuros equipos, se recomienda probar primero todos los pasos manualmente, antes de codificarlos en una receta. Esto ayuda a entender mejor los comandos, rutas y posibles errores.
* Evitar el uso de comandos como sudo dentro de las recetas Chef, ya que Chef se ejecuta como root.
* Siempre validar rutas, usuarios y permisos cuando se utilizan servicios como systemd.
* Aprovechar los recursos oficiales de documentación de Chef y los foros de la comunidad cuando surgen problemas específicos.
* Tomar capturas de pantalla durante cada etapa para no olvidar pasos importantes y facilitar la documentación.