

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
Actualización del sistema	4
Descarga del instalador de Chef Workstation	4
Instalación del paquete descargado.....	5
Verificación de la instalación	6
Verificación del binario y estructura interna.....	6
Generación de estructura base para trabajo con recetas.....	7
Cookbooks	8
Cookbook 1: fastapi_app.....	8
Paso 1: Crear el cookbook	8
Paso 2: Crear el archivo de plantilla main.py	9
Paso 3: Editar la receta default.rb	10
Paso 4: Ejecución del cookbook y validación del despliegue	13
Cookbook 2: Instalación de Ngrok.....	16
Paso 1: Crear el cookbook	16
Paso 2: Editar la receta default.rb	17
Paso 3: Ejecutar el cookbook.....	17
Paso 4: Lanzar el servidor local y script de Ngrok	17
Pruebas automatizadas con InSpec.....	17
Cookbook 3: Instalación y despliegue de Jenkins automatizado	20
Paso 1: Crear el cookbook	20
Paso 2: Editar la receta default.rb	20
Paso 3: Ejecutar la receta	22
Paso 4: Validar Jenkins en el navegador.....	23
Cookbook 4: WordPress	24
Paso 1: Crear el cookbook	24
Paso 2: Editar la receta default.rb	25
Paso 3: Ejecutar el cookbook.....	26
Paso 4: Validación del despliegue	26

Cookbook 5: RunSpringbootAPP	28
Paso 1: Crear el cookbook	28
Paso 2: Seleccionar el repositorio de ejemplo	28
Paso 3: Editar la receta default.rb	29
Paso 4: Ejecutar el cookbook.....	29
Paso 5: Validación en el navegador	30
Repositorio con los cookbooks	31
Tabla de valoración.....	32
Reflexión sobre los desafíos enfrentados	33
Conclusiones	34
Recomendaciones.....	35

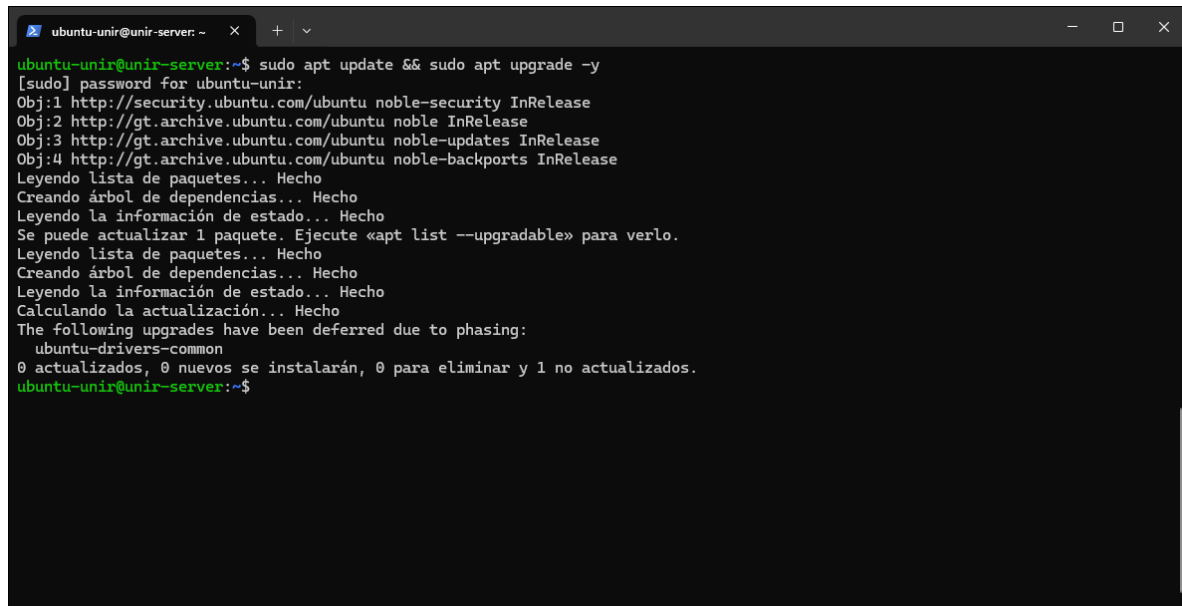
Instalación y configuración de Chef Workstation

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
```



```
ubuntu-unir@unir-server: ~  
ubuntu-unir@unir-server:~$ sudo apt update && sudo apt upgrade -y  
[sudo] password for ubuntu-unir:  
Obj:1 http://security.ubuntu.com/ubuntu noble-security InRelease  
Obj:2 http://gt.archive.ubuntu.com/ubuntu noble InRelease  
Obj:3 http://gt.archive.ubuntu.com/ubuntu noble-updates InRelease  
Obj:4 http://gt.archive.ubuntu.com/ubuntu noble-backports InRelease  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Se puede actualizar 1 paquete. Ejecute «apt list --upgradable» para verlo.  
Leyendo lista de paquetes... Hecho  
Creando árbol de dependencias... Hecho  
Leyendo la información de estado... Hecho  
Calculando la actualización... Hecho  
The following upgrades have been deferred due to phasing:  
  ubuntu-drivers-common  
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 1 no actualizados.  
ubuntu-unir@unir-server:~$
```

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
```

```
ubuntu-unir@unir-server: ~  
ubuntu-unir@unir-server:~$ wget https://packages.chef.io/files/stable/chef-workstation/24.12.1073/ubuntu/20.04/chef-workstation_24.12.1073-1_amd64.deb  
--2025-07-13 02:27:08-- https://packages.chef.io/files/stable/chef-workstation/24.12.1073/ubuntu/20.04/chef-workstation_24.12.1073-1_amd64.deb  
Resolving packages.chef.io (packages.chef.io)... 146.75.126.110  
Connecting to packages.chef.io (packages.chef.io)|146.75.126.110|:443... connected.  
HTTP request sent, awaiting response... 200 OK  
Length: 225237260 (215M) [application/x-debian-package]  
Saving to: 'chef-workstation_24.12.1073-1_amd64.deb'  
  
chef-workstation_24.12.1073-1_a 100%[=====] 214,80M 20,5MB/s in 1m 48s  
  
2025-07-13 02:28:57 (1,98 MB/s) - 'chef-workstation_24.12.1073-1_amd64.deb' saved [225237260/225237260]  
  
ubuntu-unir@unir-server:~$ ls -l  
total 219968  
-rw-rw-r-- 1 ubuntu-unir ubuntu-unir 225237260 dic 16 2024 chef-workstation_24.12.1073-1_amd64.deb  
drwxrwxr-x 4 ubuntu-unir ubuntu-unir 4096 jul 5 23:59 web-app  
ubuntu-unir@unir-server:~$  
ubuntu-unir@unir-server:~$ ls -lh  
total 215M  
-rw-rw-r-- 1 ubuntu-unir ubuntu-unir 215M dic 16 2024 chef-workstation_24.12.1073-1_amd64.deb  
drwxrwxr-x 4 ubuntu-unir ubuntu-unir 4,0K jul 5 23:59 web-app  
ubuntu-unir@unir-server:~$
```

Instalación del paquete descargado

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

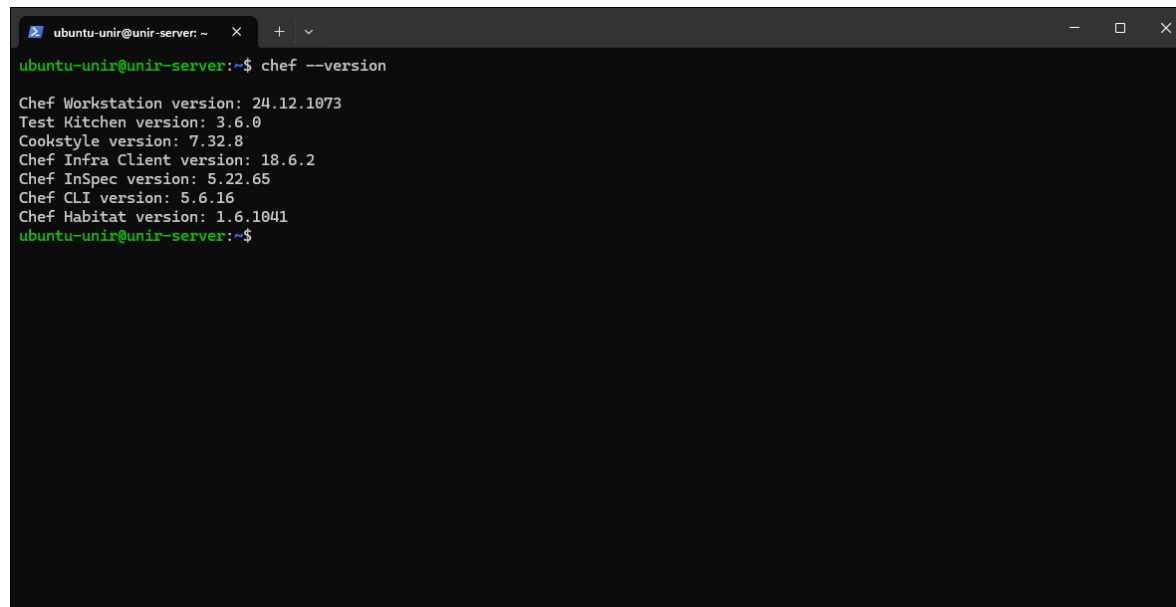
```
sudo dpkg -i chef-workstation_24.12.1073-1_amd64.deb
```

```
ubuntu-unir@unir-server: ~  
ubuntu-unir@unir-server:~$ sudo dpkg -i chef-workstation_24.12.1073-1_amd64.deb  
[sudo] password for ubuntu-unir:  
Seleccionando el paquete chef-workstation previamente no seleccionado.  
(Leyendo la base de datos ... 86975 ficheros o directorios instalados actualmente.)  
Preparando para desempaquetar chef-workstation_24.12.1073-1_amd64.deb ...  
Desempaquetando chef-workstation (24.12.1073-1) ...  
Configurando chef-workstation (24.12.1073-1) ...  
  
Chef Workstation ships with a toolbar application, the Chef Workstation App.  
To run this application some additional dependencies must be installed.  
Using your platform's package manager to install the 'electron' package is  
the easiest way to meet the dependency requirements.  
  
You can then launch the App by running 'chef-workstation-app'.  
The App will then be available in the system tray.  
  
Thank you for installing Chef Workstation!  
You can find some tips on getting started at https://docs.chef.io/workstation/getting\_started/  
  
ubuntu-unir@unir-server:~$ |
```

Verificación de la instalación

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

```
chef --version
```

A terminal window titled 'ubuntu-unir@unir-server: ~' with a dark background. The command 'chef --version' has been entered and executed. The output lists the versions of various Chef components: Chef Workstation (24.12.1073), Test Kitchen (3.6.0), Cookstyle (7.32.8), Chef Infra Client (18.6.2), Chef InSpec (5.22.65), Chef CLI (5.6.16), and Chef Habitat (1.6.1041).

```
ubuntu-unir@unir-server:~$ chef --version
Chef Workstation version: 24.12.1073
Test Kitchen version: 3.6.0
Cookstyle version: 7.32.8
Chef Infra Client version: 18.6.2
Chef InSpec version: 5.22.65
Chef CLI version: 5.6.16
Chef Habitat version: 1.6.1041
ubuntu-unir@unir-server:~$
```

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.

```
ubuntu-unir@unir-server: ~$ which chef
/usr/bin/chef
ubuntu-unir@unir-server:~$ ls /opt/chef-workstation/bin/
berks      chef-client  chef-solo      delivery  kitchen      stove
chef        chef-resource-inspector  chef-vault    fauxhai   knife
chef-analyze  chef-run      chef-windows-service  foodcritic  mixlib-install
chef-apply    chef-service-manager  chef-zero     hab        ohai
chef-cli      chef-shell      cookstyle     inspect   ruby-env-script.rb
ubuntu-unir@unir-server:~$
```

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.

```
ubuntu-unir@unir-server: ~/ $ chef generate repo chef-repo
Generating Chef Infra repo chef-repo
- Ensuring correct Chef Infra repo file content

Your new Chef Infra repo is ready! Type 'cd chef-repo' to enter it.
ubuntu-unir@unir-server:~$ cd chef-repo/
ubuntu-unir@unir-server:~/chef-repo$ ls
chefignore  cookbooks  data_bags  LICENSE  policyfiles  README.md
ubuntu-unir@unir-server:~/chef-repo$
```

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

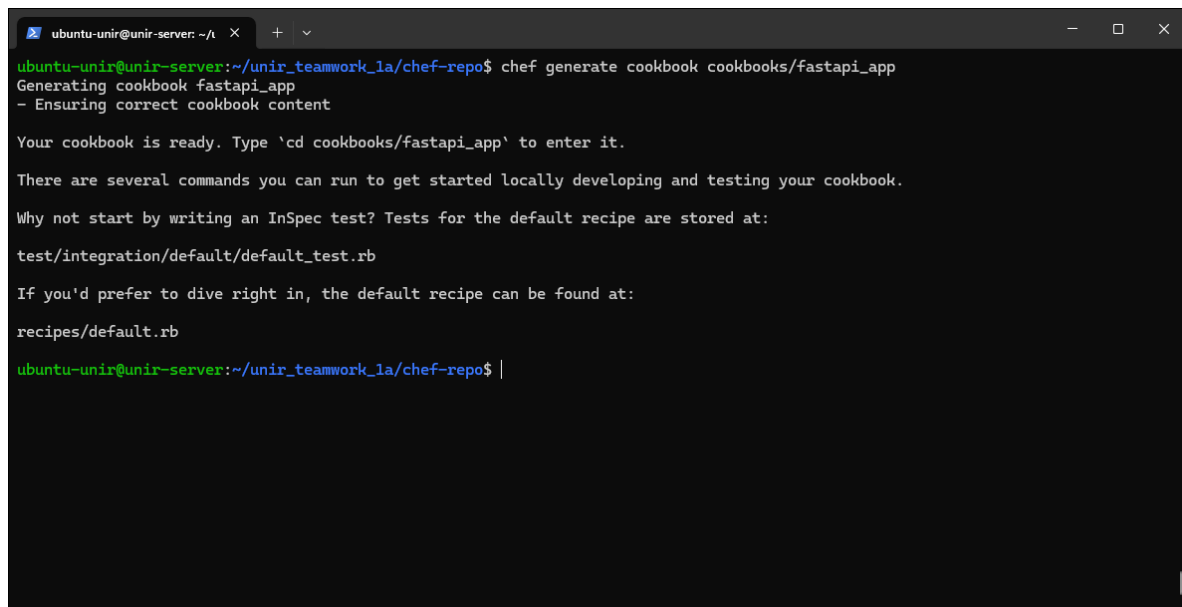
Cookbook personalizado llamado fastapi_app con el objetivo de automatizar el despliegue de una aplicación web FastAPI sobre un servidor Ubuntu. El proceso incluyó:

- 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
```



```
ubuntu-unir@unir-server: ~/unir_teamwork_1a/chef-repo$ chef generate cookbook cookbooks/fastapi_app
Generating cookbook fastapi_app
- Ensuring correct cookbook content

Your cookbook is ready. Type 'cd cookbooks/fastapi_app' to enter it.

There are several commands you can run to get started locally developing and testing your cookbook.

Why not start by writing an InSpec test? Tests for the default recipe are stored at:
test/integration/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:
recipes/default.rb

ubuntu-unir@unir-server:~/unir_teamwork_1a/chef-repo$ |
```

Este comando generó la estructura básica del cookbook *fastapi_app*.


```
ubuntu-unir@unir-server: ~/t  X + v
ubuntu-unir@unir-server:~/unir_teamwork_1a/chef-repo$ cd cookbooks/fastapi_app/
ubuntu-unir@unir-server:~/unir_teamwork_1a/chef-repo/cookbooks/fastapi_app$ ls -R
.:
CHANGELOG.md  cheffignore  compliance  kitchen.yml  LICENSE  metadata.rb  Policyfile.rb  README.md  recipes  test

./compliance:
inputs  profiles  README.md  waivers

./compliance/inputs:

./compliance/profiles:

./compliance/waivers:

./recipes:
default.rb

./test:
integration

./test/integration:
default

./test/integration/default:
default_test.rb
ubuntu-unir@unir-server:~/unir_teamwork_1a/chef-repo/cookbooks/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"
    }

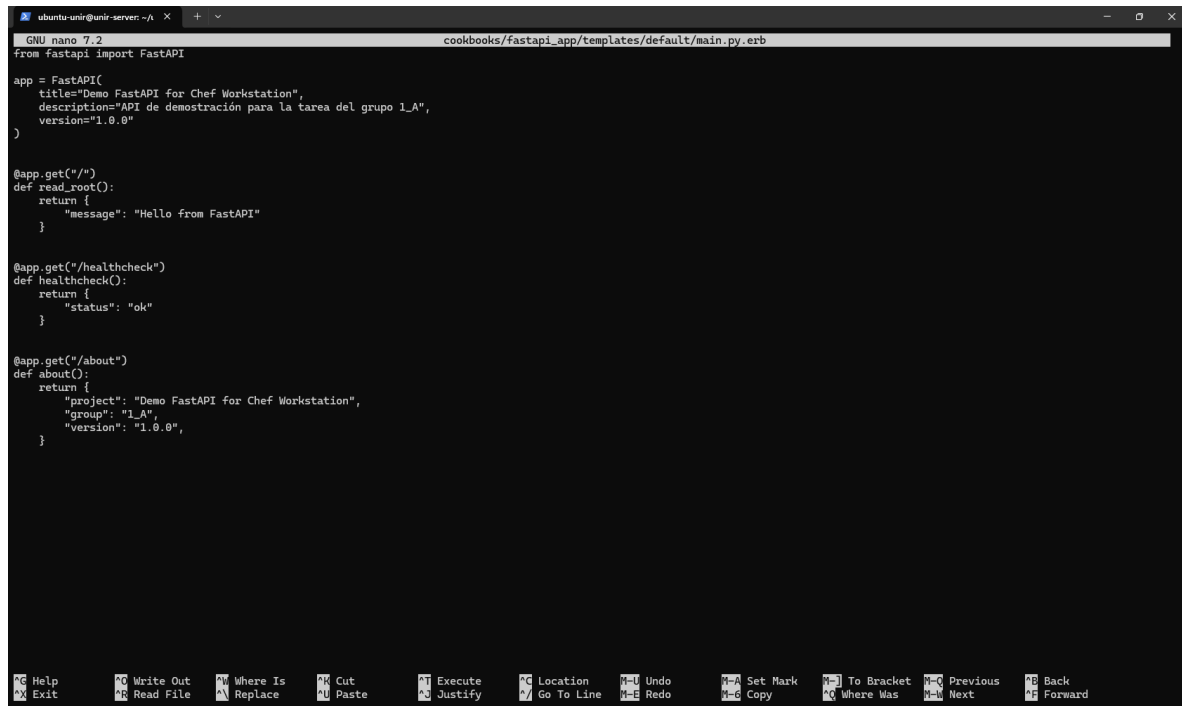
@app.get("/about")
def about():
    return {
```

```

    "project": "Demo FastAPI for Chef Workstation",
    "group": "1_A",
    "version": "1.0.0",
  }

```

Con el siguiente contenido:



```

GNU nano 7.2 cookbooks/fastapi_app/templates/default/main.py.erb
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",
    }

```

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.

```
ubuntu-unir@unir-server: ~/A  +  x
GNU nano 7.2 cookbook/fastapi_app/recipes/default.rb
#
# Cookbook:: fastapi_app
# Recipe:: default
#
# Actualización del sistema (opcional pero recomendado)
execute 'apt_update' do
  command 'apt update && apt upgrade -y'
end

# Instalación de herramientas necesarias
package %([python3-venv curl]) do
  action :install
end

# Crear directorio para la aplicación
directory '/home/ubuntu-unir/fastapi_app' do
  owner 'ubuntu-unir'
  group 'ubuntu-unir'
  mode '0755'
  recursive true
  action :create
end

# Crear entorno virtual solo si no existe
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

# Instalar FastAPI en el entorno virtual
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

# Crear el archivo main.py desde plantilla
template '/home/ubuntu-unir/fastapi_app/main.py' do
```

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.

```
ubuntu-unir@unir-server: ~$ +
```

```
ubuntu-unir@unir-server:/unir/teamwork_la/chef-repo$ sudo chef-client --local-mode --runlist 'recipe[fastapi_app]'
[2025-07-13T16:46:02+0800] WARN: No config file found or specified on command line. Using command line options instead.
Using Chef Client version 19.8.2
Patents: https://www.chef.io/patents
Infra-Pulse starting
Resolving cookbooks for run list: ["fastapi_app"]
Synchronizing cookbook:
  - fastapi_app (0.1.0)
Installing cookbook gem dependencies:
Compiling cookbooks...
Loading Chef InSpec profile files:
Loading Chef InSpec input files:
Loading Chef InSpec waiver files:
Converting 0 resources
Recipe: fastapi_app::default
 * execute[create_repo] action run
   * node["*"]::execute["mkdir -p"]
   * node["*"]::execute["python3 -m pip install --no-cache-dir fastapi"]
   * directory[/home/ubuntu-unir/fastapi_app] action create
   * change mode from "" to "755"
   * change owner from "" to ubuntu-unir
   * change group from "" to ubuntu-unir
 * execute[create_virtualenv] action run
   * node["*"]::execute["python3 -m venv /home/ubuntu-unir/.venv"]
 * execute[install_fastapi] action run
   * template[/home/ubuntu-unir/fastapi_app/main.py] action create
     * template[/home/ubuntu-unir/fastapi_app/main.py] [template]
       --- /home/ubuntu-unir/fastapi_app/main.py 2025-07-13 16:48:36.460567916 +0800
       +++ /home/ubuntu-unir/fastapi_app/main.py 2025-07-13 16:48:36.460567916 +0800
       @@ -1,13 @@
       +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",
       +    }
       +
       +--- change mode from "" to "755"
       +--- change owner from "" to ubuntu-unir
       +--- change group from "" to ubuntu-unir
 * file[/etc/systemd/system/fastapi.service] action create (up to date)
 * execute[reload_systemd] action run
   * service[fastapi] reload
 * service[fastapi] action enable (up to date)
 * service[fastapi] action start (up to date)
```

```
Running handlers:
Running handlers complete
Infra-Pulse completed. All resources updated in 82 minutes 56 seconds
ubuntu-unir@unir-server:/unir/teamwork_la/chef-repo$
```

Verificar que el servicio esté activo

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

```
sudo systemctl status fastapi
```

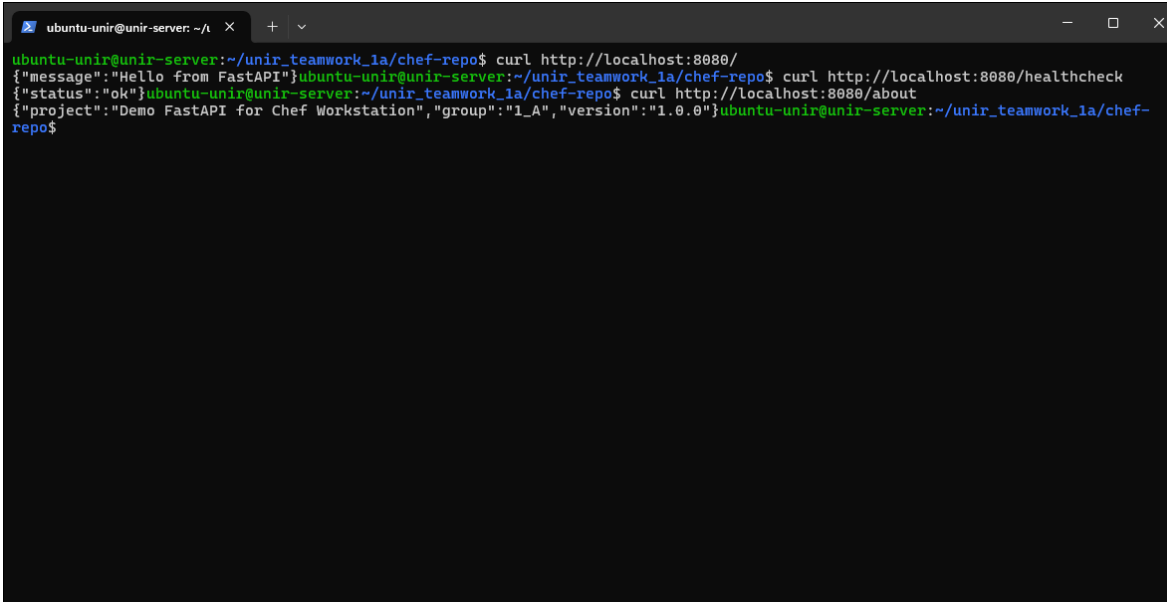
```
ubuntu-unir@unir-server: ~/l  X + -
ubuntu-unir@unir-server:~/unir_teamwork_1a/chef-repo$ sudo systemctl status fastapi
● fastapi.service - FastAPI Application
   Loaded: loaded (/etc/systemd/system/fastapi.service; enabled; preset: enabled)
   Active: active (running) since Sun 2025-07-13 15:56:27 UTC; 54min ago
     Main PID: 658 (python)
        Tasks: 6 (limit: 2268)
       Memory: 62.0M (peak: 73.4M)
          CPU: 13.321s
       CGroup: /system.slice/fastapi.service
               └─658 /home/ubuntu-unir/fastapi_app/.venv/bin/python -m fastapi run main.py --host 0.0.0.0 --port 8080

Jul 13 15:56:40 unir-server python[658]:
Jul 13 15:56:40 unir-server python[658]:      server  Server started at http://0.0.0.0:8080
Jul 13 15:56:40 unir-server python[658]:      server  Documentation at http://0.0.0.0:8080/docs
Jul 13 15:56:40 unir-server python[658]:
Jul 13 15:56:40 unir-server python[658]:      Logs:
Jul 13 15:56:40 unir-server python[658]:
Jul 13 15:56:40 unir-server python[658]:      INFO    Started server process [658]
Jul 13 15:56:40 unir-server python[658]:      INFO    Waiting for application startup.
Jul 13 15:56:40 unir-server python[658]:      INFO    Application startup complete.
Jul 13 15:56:40 unir-server python[658]:      INFO    Uvicorn running on http://0.0.0.0:8080 (Press CTRL+C to quit)
ubuntu-unir@unir-server:~/unir_teamwork_1a/chef-repo$
```

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
```

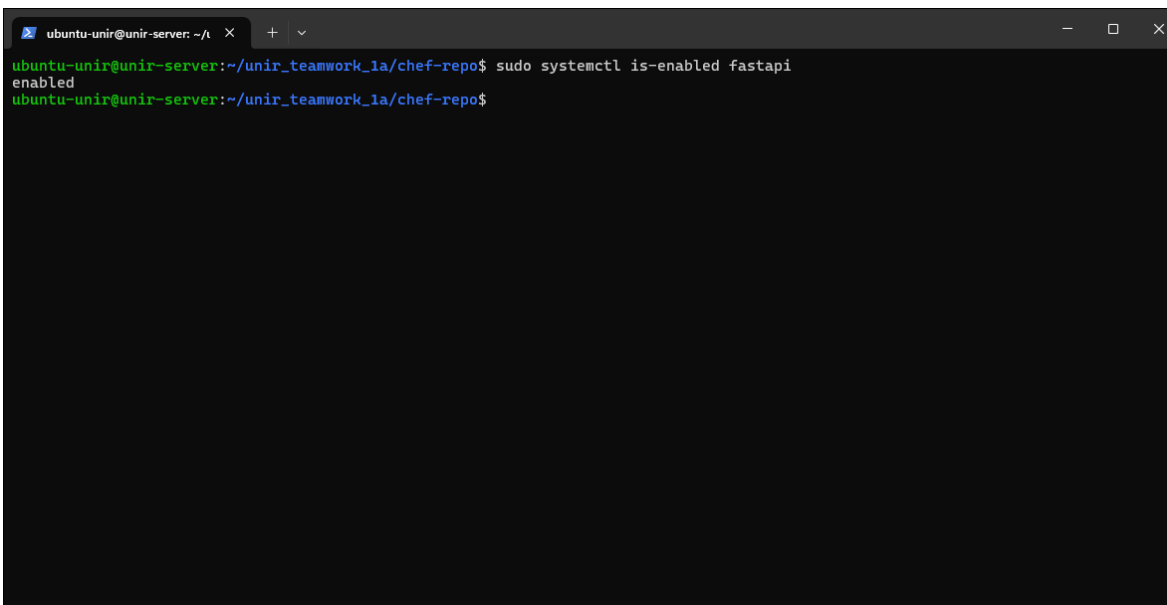


```
ubuntu-unir@unir-server: ~/unir_teamwork_1a/chef-repo$ curl http://localhost:8080/  
{  
  "message": "Hello from FastAPI"  
}  
ubuntu-unir@unir-server: ~/unir_teamwork_1a/chef-repo$ curl http://localhost:8080/healthcheck  
{  
  "status": "ok"  
}  
ubuntu-unir@unir-server: ~/unir_teamwork_1a/chef-repo$ curl http://localhost:8080/about  
{  
  "project": "Demo FastAPI for Chef Workstation",  
  "group": "1_A",  
  "version": "1.0.0"  
}
```

Verificar inicio automático del servicio

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

```
sudo systemctl is-enabled fastapi
```



```
ubuntu-unir@unir-server: ~/unir_teamwork_1a/chef-repo$ sudo systemctl is-enabled fastapi  
enabled  
ubuntu-unir@unir-server: ~/unir_teamwork_1a/chef-repo$
```

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:

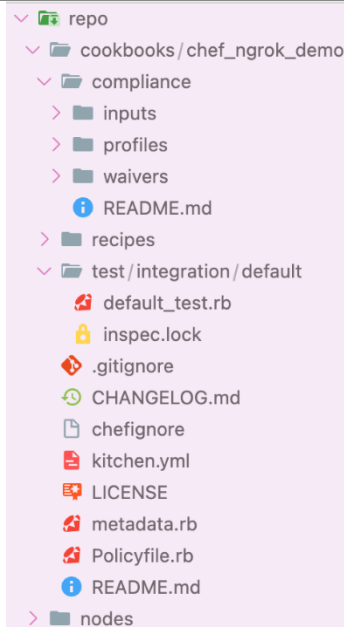
- Descarga del binario ARM64 de Ngrok.
- Creación del archivo de configuración `ngrok.yml` con token de autenticación.
- Generación de un script de lanzamiento para abrir túneles hacia puertos locales.
- Validación del entorno mediante pruebas automatizadas con InSpec.

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
```



```

repo
├── cookbooks
│   ├── chef_ngrok_demo
│   ├── compliance
│   ├── inputs
│   ├── profiles
│   ├── waivers
│   ├── README.md
│   ├── recipes
│   └── test/integration/default
│       ├── default_test.rb
│       ├── inspec.lock
│       ├── .gitignore
│       ├── CHANGELOG.md
│       ├── chefignore
│       ├── kitchen.yml
│       ├── LICENSE
│       ├── metadata.rb
│       ├── Policyfile.rb
│       └── README.md
└── nodes

```


Paso 2: Editar la receta default.rb

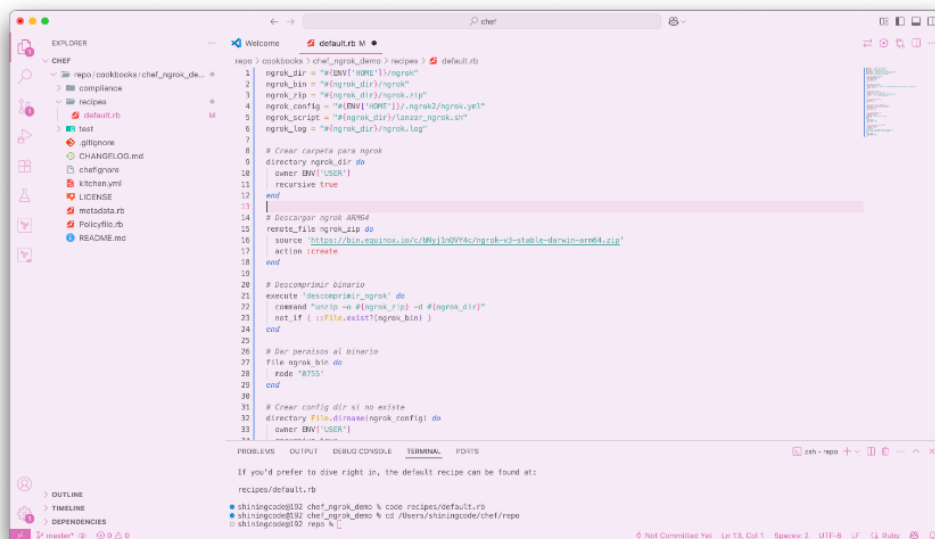
Se editó el archivo **recipes/default.rb** para automatizar los siguientes procesos:

- 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]'
```



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:

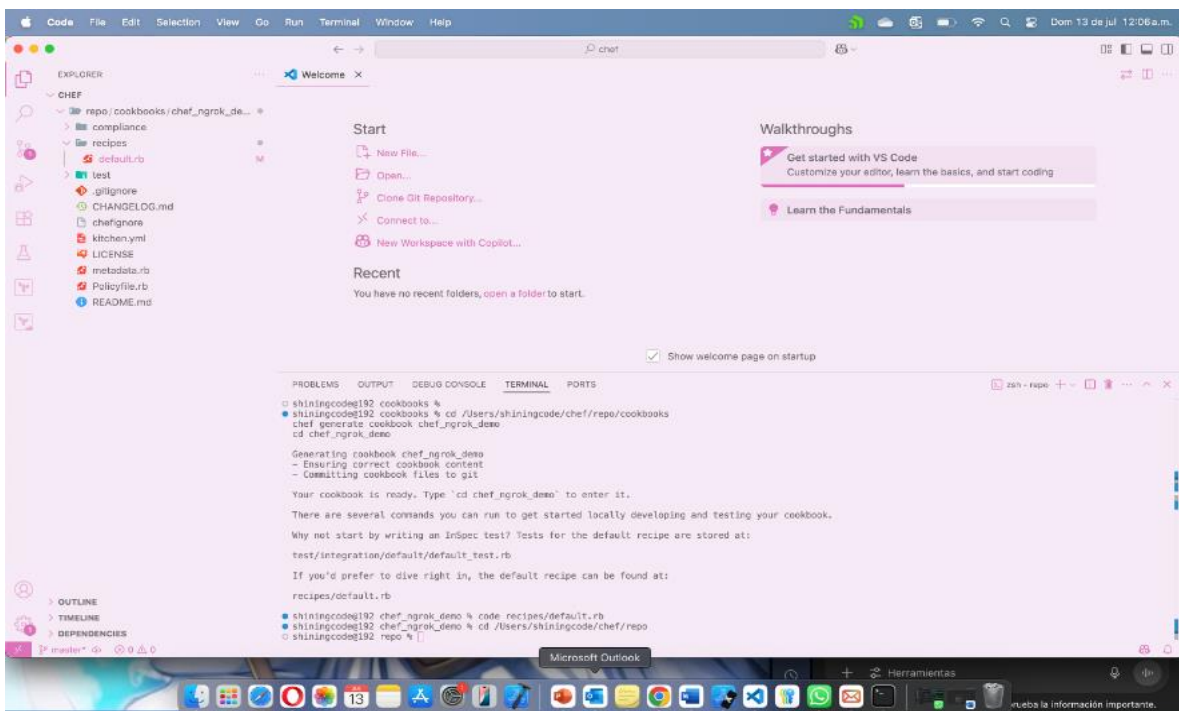
```
test/integration/default/default_test.rb
```

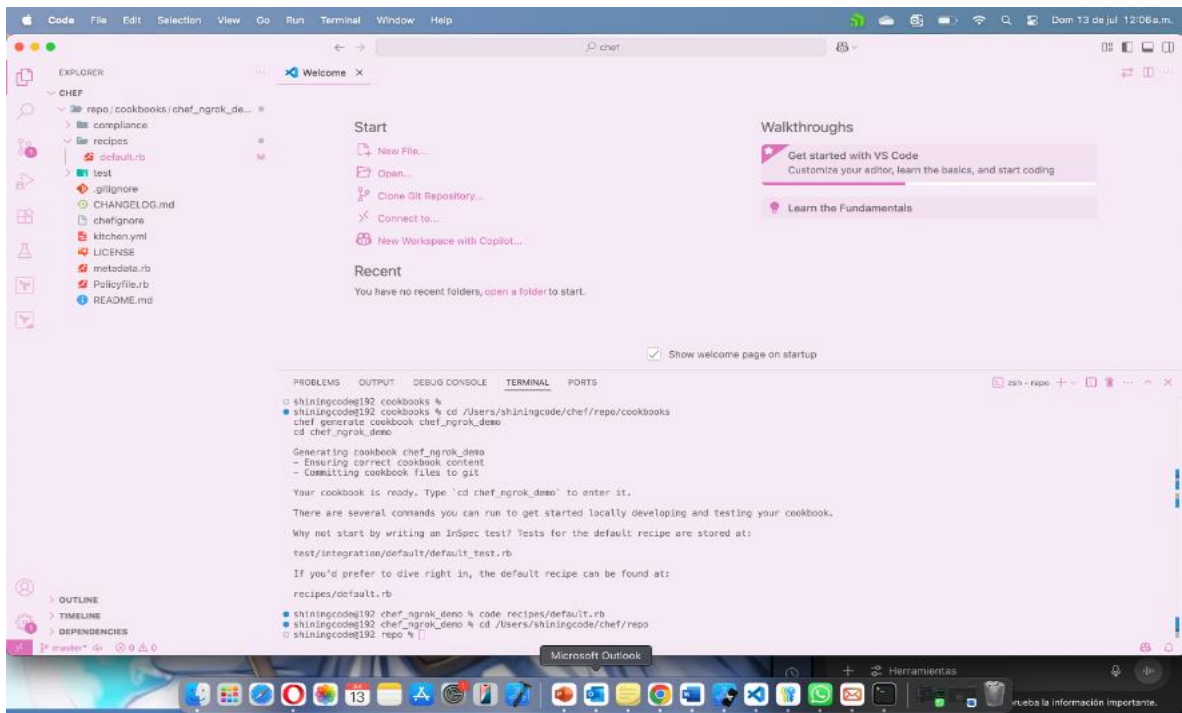
Código de prueba:

```
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:

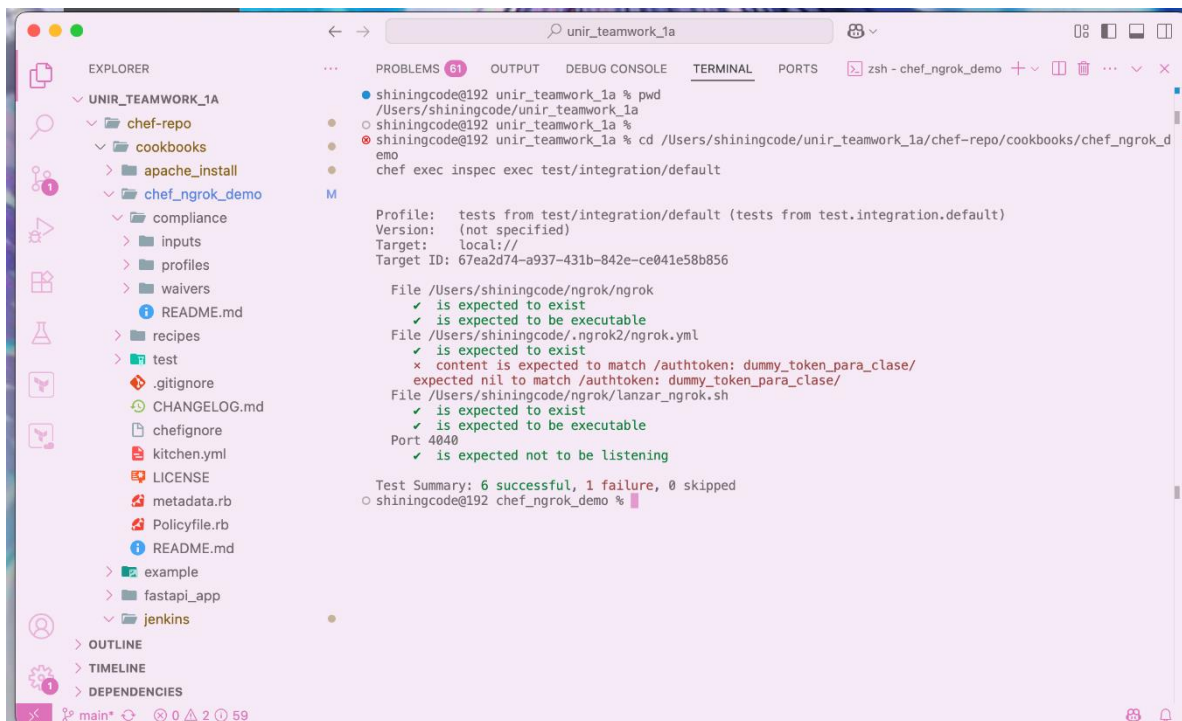
- El binario fue descargado correctamente.
- El archivo de configuración existe y contiene el token esperado.
- El script de ejecución es funcional.
- El puerto 4040 (dashboard web) no está escuchando por defecto.





Ejecución:

chef exec inspec exec test/integration/default



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 borraría. 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.

```
- execute      sudo docker stop jenkins || true
               sudo docker rm jenkins || true

* execute[run_jenkins_container] action run
- execute      puts "ejecutando docker-jenkins"
               docker run -d --name jenkins      -p 8080:8080      jenkins/jenkins:lts

* execute[wait_for_jenkins_container] action run
- execute      while ! docker exec jenkins test -f /var/jenkins_home/secrets/i
nitialAdminPassword; do
               sleep 2
               done

* ruby_block[print_initial_admin_password] action run
Jenkins initial admin password: b8216f6c40ae45e4b12891290f8ced59

- execute the ruby block print_initial_admin_password

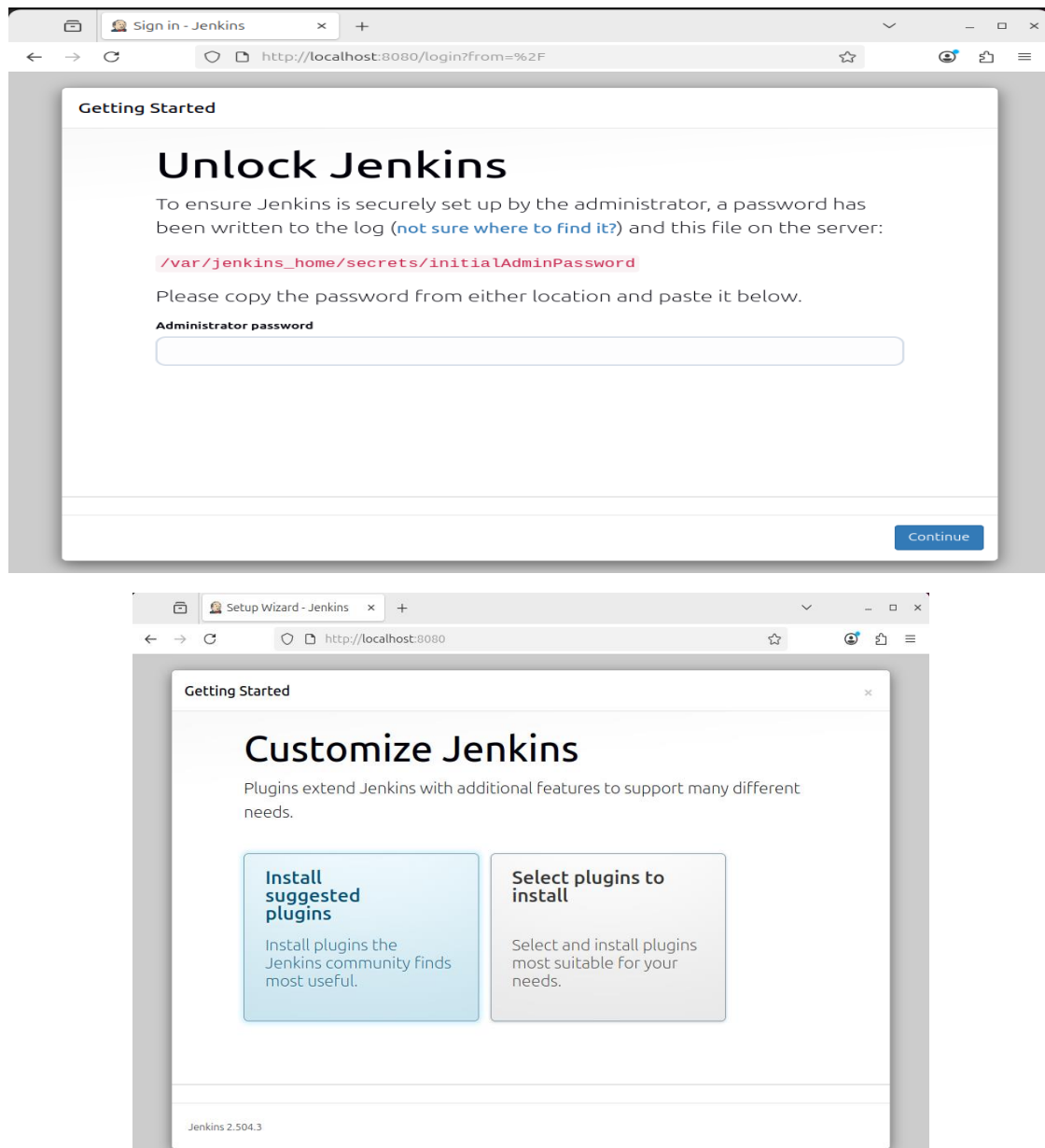
Running handlers:
Running handlers complete
Chef Infra Client finished, 5/6 resources updated in 13 seconds
dtrejo@maestria:~/git_repo/unir_teamwork_1a/Cheff_configuration/workspace/chef-r
epo$ sudo chef-client --local-mode --runlist 'recipe[jenkins]' --config-option c
ookbook_path="$(pwd)/cookbooks"$
```

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:

```
jlopzz@Gerardo:~/unir_teamwork_1a$ cd chef-repo/cookbooks/
jlopzz@Gerardo:~/unir_teamwork_1a/chef-repo/cookbooks$ chef generate cookbook wordpress

[2025-07-18T09:00:26-06:00] WARN: Please install an English UTF-8 locale for Chef Infra Client to
use, falling back to C locale and disabling UTF-8 support.
Generating cookbook wordpress
- Ensuring correct cookbook content

Your cookbook is ready. Type 'cd wordpress' to enter it.

There are several commands you can run to get started locally developing and testing your cookbook
.

Why not start by writing an InSpec test? Tests for the default recipe are stored at:
test/integration/default/default_test.rb

If you'd prefer to dive right in, the default recipe can be found at:
recipes/default.rb
```


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:

```
jlopzz@Gerardo:~/unir_teamwork_1a/chef-repo/cookbooks/wordpress$ code recipes/default.rb
jlopzz@Gerardo:~/unir_teamwork_1a/chef-repo/cookbooks/wordpress$ |
```

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(phi 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'
  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:

```
jlopzz@Gerardo:~/unir_teamwork_1a/chef-repo/cookbooks/wordpress$ sudo chef-client --local-mode --r
unlist 'recipe[wordpress]'
[2025-07-18T09:50:50-06:00] WARN: No config file found or specified on command line. Using command
line options instead.
Chef Infra Client, version 18.6.2
Patents: https://www.chef.io/patents
Infra Phase starting
[2025-07-18T09:50:52-06:00] ERROR: shard_seed: Failed to get dmi property serial_number: is dmidec
ode installed?
Resolving cookbooks for run list: ["wordpress"]
Synchronizing cookbooks:
- wordpress (0.1.0)
Installing cookbook gem dependencies:
Compiling cookbooks...
Loading Chef InSpec profile files:
Loading Chef InSpec input files:
Loading Chef InSpec waiver files:
Converging 10 resources
Recipe: wordpress::default
* apt_package[apache2] action install (up to date)
* service[apache2] action enable (up to date)
* service[apache2] action start (up to date)
```

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:

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

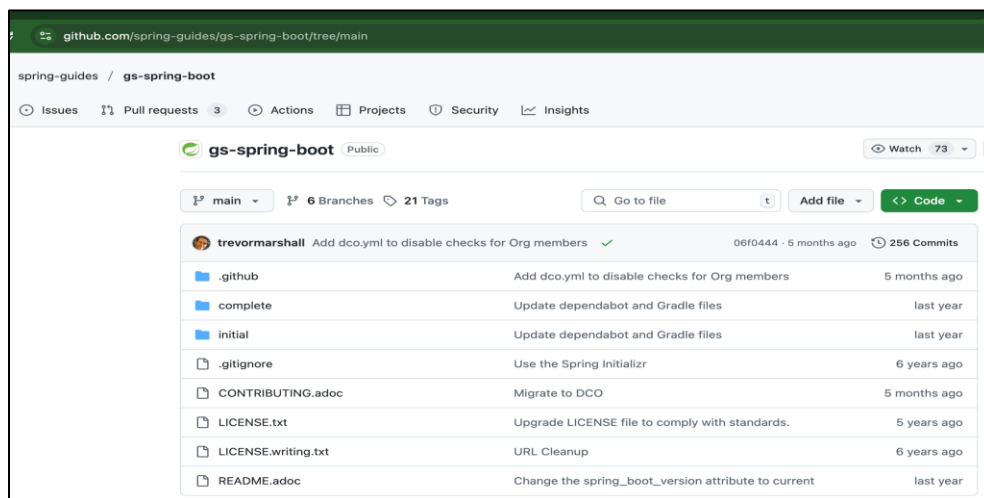
```
chef generate cookbook cookbooks/RunSpringbootAppCookbook 3
```

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!".



Paso 3: Editar la receta default.rb

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

```
chef-repo > cookbooks > runSpringbootAPP > recipes > default.rb
You, 24 minutes ago | 1 author (You)
1 #Receta que instala dependencias (Solo MacOS), clona un repositorio, construye un jar y corre un aplicacion de java
2 #Se define el url del repositorio publico de github
3 REPO_URL = "https://github.com/spring-guides/gs-spring-boot.git"
4 #Se define como se llamara la app de springboot
5 APP_NAME = "your-springboot-api"
6 #Se define el directorio donde se clonara el repo
7 APP_DIR = "#{ENV['HOME']}/#{APP_NAME}"
8 #Se define el sub-directorio donde se encuentra el archivo pom.xml dentro del repositorio (en el caso del repositorio de ejemplo, se encuentra en /complete/)
9 APP_SUBDIR = "#{APP_DIR}/complete"
10
11
12 #Instalamos las dependencias usando brew
13 homebrew_package 'git' do
14   action :install
15 end
16 homebrew_package 'openjdk' do
17   action :install
18 end
19 homebrew_package 'maven' do
20   action :install
21 end
22
23
24 # Clonamos el repositorio de la rama main
25 git APP_DIR do
26   repository REPO_URL
27   revision 'main'
28   action :sync
29 end
30
31 # Construimos el JAR usando maven
32 execute 'build spring boot app' do
33   cwd APP_SUBDIR
34   command 'mvn clean package -DskipTests'
35 end
36
37 #Eliminamos un proceso de java corriendo en caso de que exista uno
38 execute 'kill existing spring boot app' do
39   command "pkill -f 'java -jar' || true"
40   only_if "pgrep -f 'java -jar'"
41 end
42
43 # Corremos la aplicacion de java
44 execute 'run spring boot app' do
45   cwd APP_SUBDIR
46   command "nohup java -jar target/*.jar"
47 end
You, 24 minutes ago • Feat: Se agrega cookbook RunSpringbootAPP ...
```

Paso 4: Ejecutar el cookbook

Una vez completada la receta, se ejecutó con:

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

```
coppel@192 chef-repo % sudo chef-client --local-mode --runlist 'recipe[runSpringbootAP]'
```

```
[2025-07-13T16:54:03-07:00] WARN: No config file found or specified on command line. Using command line options instead.
Chef Infra Client, version 18.7.10
Patents: https://www.chef.io/patents
Infra Phase starting
Resolving cookbooks for run list: ["runSpringbootAP"]
Synchronizing cookbooks:
  - runSpringbootAP (0.1.0)
Installing cookbook gem dependencies:
Compiling cookbooks...
Loading Chef InSpec profile files:
Loading Chef InSpec input files:
Loading Chef InSpec waiver files:
Converging 7 resources
Recipe: runSpringbootAP::default
  * homebrew_package[git] action install (up to date)
  * homebrew_package[openjdk] action install (up to date)
  * homebrew_package[maven] action install (up to date)
  * git[/Users/coppel/your-springboot-ap] action sync (up to date)
  * execute[build spring boot app] action run
    - execute mvn clean package -DskipTests
  * execute[kill existing spring boot app] action run
    - execute pkill -f 'java -jar' || true
  * execute[run spring boot app] action run[C[2025-07-13T17:00:12-07:00] FATAL: SIGINT received, stopping

^C[2025-07-13T17:00:15-07:00] FATAL: SIGINT received, stopping

Running handlers:
[2025-07-13T17:00:15-07:00] ERROR: Running exception handlers
Running handlers complete
[2025-07-13T17:00:15-07:00] ERROR: Exception handlers complete
Infra Phase failed. 2 resources updated in 06 minutes 11 seconds
[2025-07-13T17:00:15-07:00] FATAL: Stacktrace dumped to /Users/coppel/.chef/local-mode-cache/cache/chef-stacktrace.out
[2025-07-13T17:00:15-07:00] FATAL:
[2025-07-13T17:00:15-07:00] FATAL: PLEASE PROVIDE THE CONTENTS OF THE stacktrace.out FILE (above) IF YOU FILE A BUG REPORT
[2025-07-13T17:00:15-07:00] FATAL:
coppel@192 chef-repo % sudo chef-client --local-mode --runlist 'recipe[runSpringbootAP]'
```

```
Password:
[2025-07-13T17:00:28-07:00] WARN: No config file found or specified on command line. Using command line options instead.
Chef Infra Client, version 18.7.10
Patents: https://www.chef.io/patents
Infra Phase starting
Resolving cookbooks for run list: ["runSpringbootAP"]
Synchronizing cookbooks:
  - runSpringbootAP (0.1.0)
Installing cookbook gem dependencies:
Compiling cookbooks...
Loading Chef InSpec profile files:
Loading Chef InSpec input files:
Loading Chef InSpec waiver files:
Converging 7 resources
Recipe: runSpringbootAP::default
  * homebrew_package[git] action install (up to date)
  * homebrew_package[openjdk] action install (up to date)
  * homebrew_package[maven] action install (up to date)
  * git[/Users/coppel/your-springboot-ap] action sync (up to date)
  * execute[build spring boot app] action run
    - execute mvn clean package -DskipTests
  * execute[kill existing spring boot app] action run
    - execute pkill -f 'java -jar' || true
  * execute[run spring boot app] action run
```

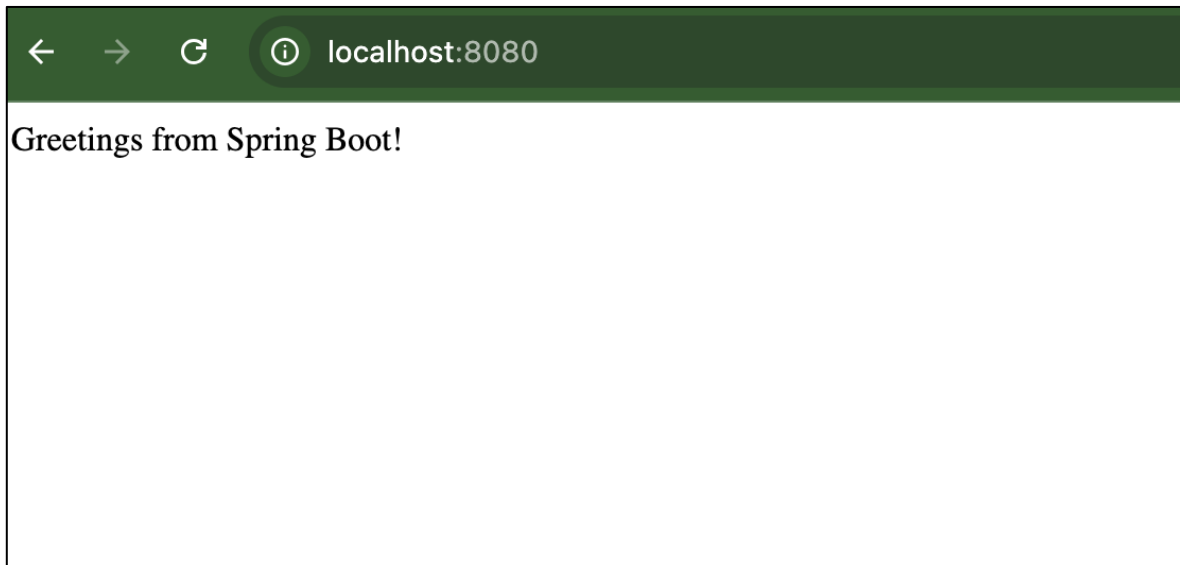
Paso 5: Validación en el navegador

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

```
http://localhost:8080
```

Y se validó la respuesta esperada:

```
Greetings from Spring Boot!
```



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.