FP.050 - (P) Devops y cloud computing

Grupo: ALT_CTRL_Supr PRODUCTO 2

Universitat Oberta de Catalunya

Descripción

• Despliegue automatizado en la nube de AWS y creación de servicio de alta disponibilidad a través de Kubernetes.

Objetivos

• Los objetivos principales del producto son:

Configurar el despliegue automatizado en el servidor de AW
Configurar el servicio en alta disponibilidad mediante la utilización de Kubernetes.

- Alvaro Parada De Lucas
- Bernat Diaz Calafat
- Rubén VIcente Gilabert

Contenido

Caso práctico

1. Leer detenidamente estas instrucciones e identificar los requerimientos de la actividad.

2. Revisar detenidamente la rúbrica de evaluación.

3. Consultar los recursos necesarios facilitados en el aula.

4. Conectarse al servidor EC2 de AWS.

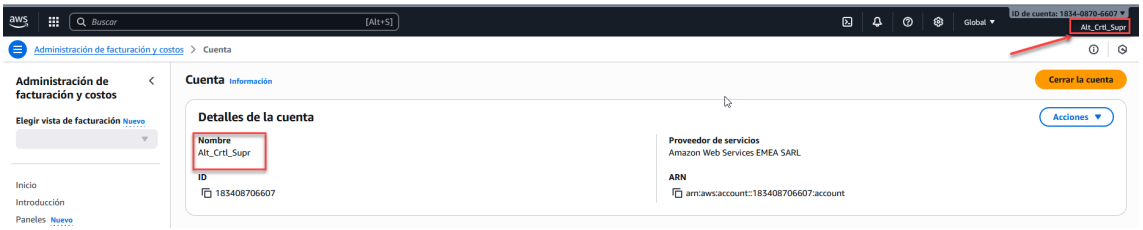
El primer paso es crear una nueva cuenta en AWS seguido de crear una nueva instancia. Deberemos tener en cuenta si queremos el servicio de pago, o si por lo contrario con el servicio gratuito ya podemos trabajar. En nuestro caso utilizaremos el servicio gratuito. Para registrar nuestra cuenta y tener nuestra instancia seguiremos estos pasos:

Primero deberemos buscar Amazon AWS para ir a la página, y seleccionar “Crear una cuenta para AWS”

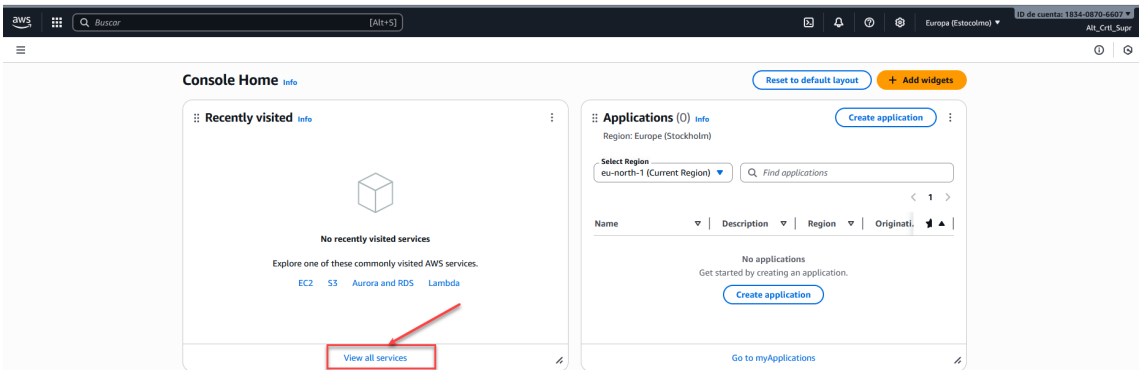
O también podemos acceder desde los recursos de aprendizaje del grado:

Una vez aquí introduciremos nuestros datos para crear la cuenta:

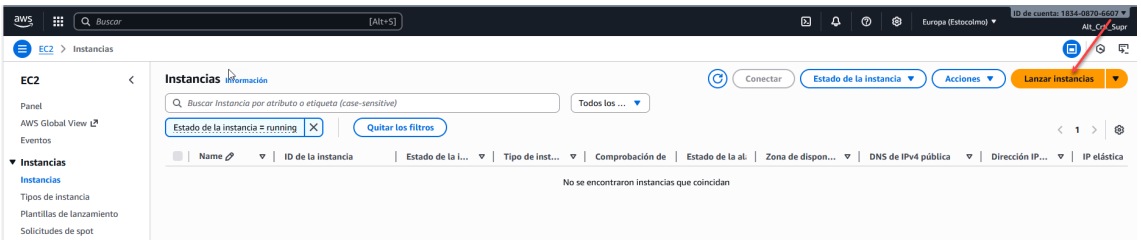
Seguiremos los pasos necesarios hasta estar dentro de la cuenta que hemos creado.



A continuación, en la página que es la principal entraremos en “Todos los servicios” > Informática > EC2



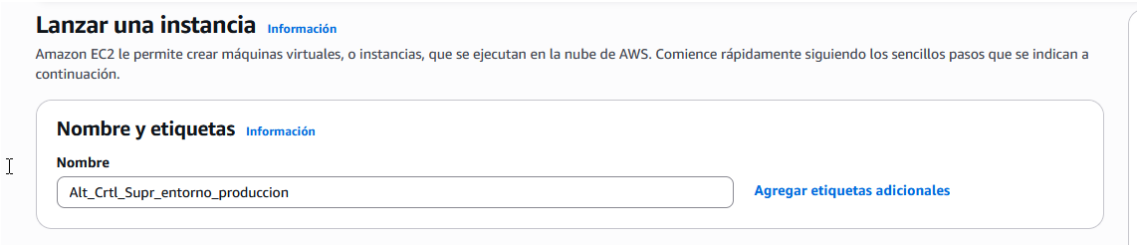
Una vez instalado, veremos que no tenemos ningún recurso, y, por lo tanto, ninguna instancia.
El siguiente paso es lanzar una instancia para poder crearla. Damos clic derecho al botón en naranja que pone “Lanzar instancias”:



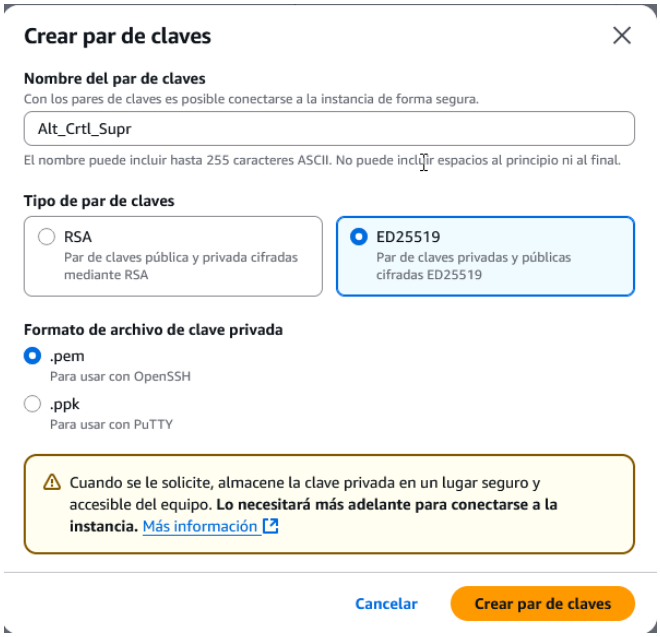
A partir de aquí configuraremos nuestra instancia. Para ello seguiremos los pasos indicados a continuación.

En nuestro caso hemos configurado la instancia de esta manera:

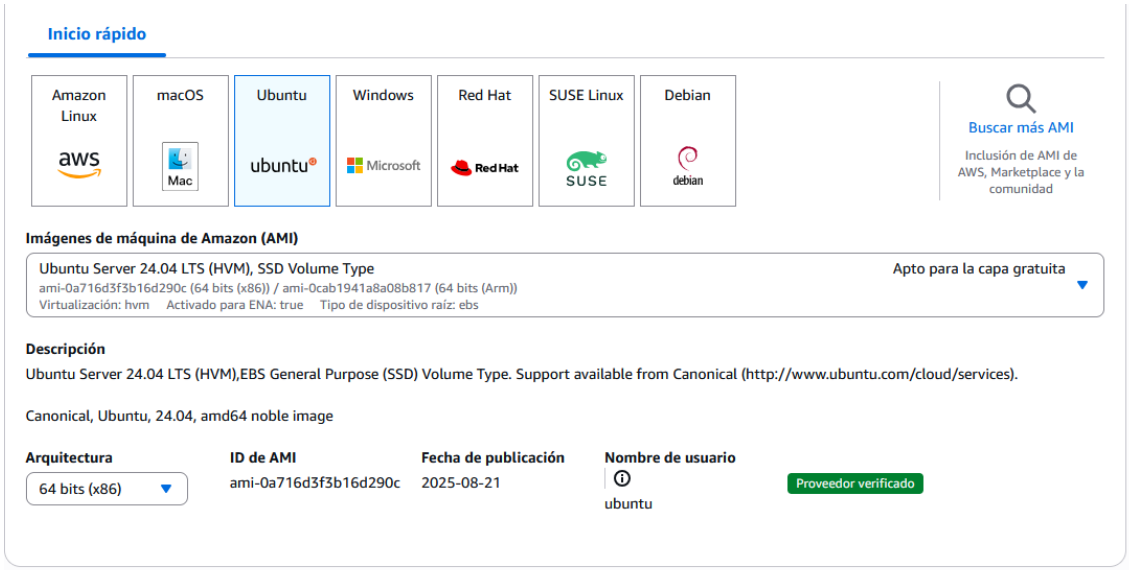
- Nombre de la instancia.



- Hemos configurado un par de claves.

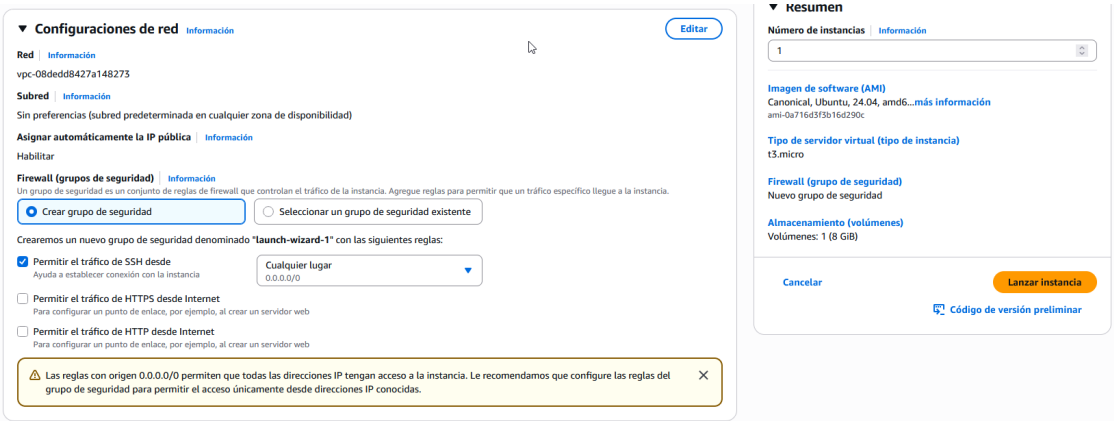


- Para el Inicio rápido, hemos elegido Ubuntu.



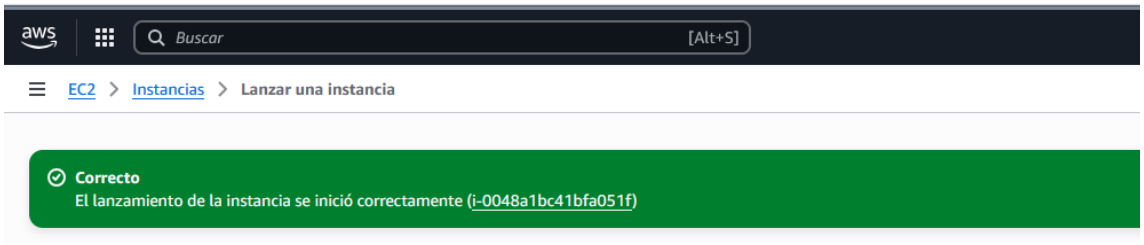
- Seleccionamos el tipo de instancia.

- Configuración de red

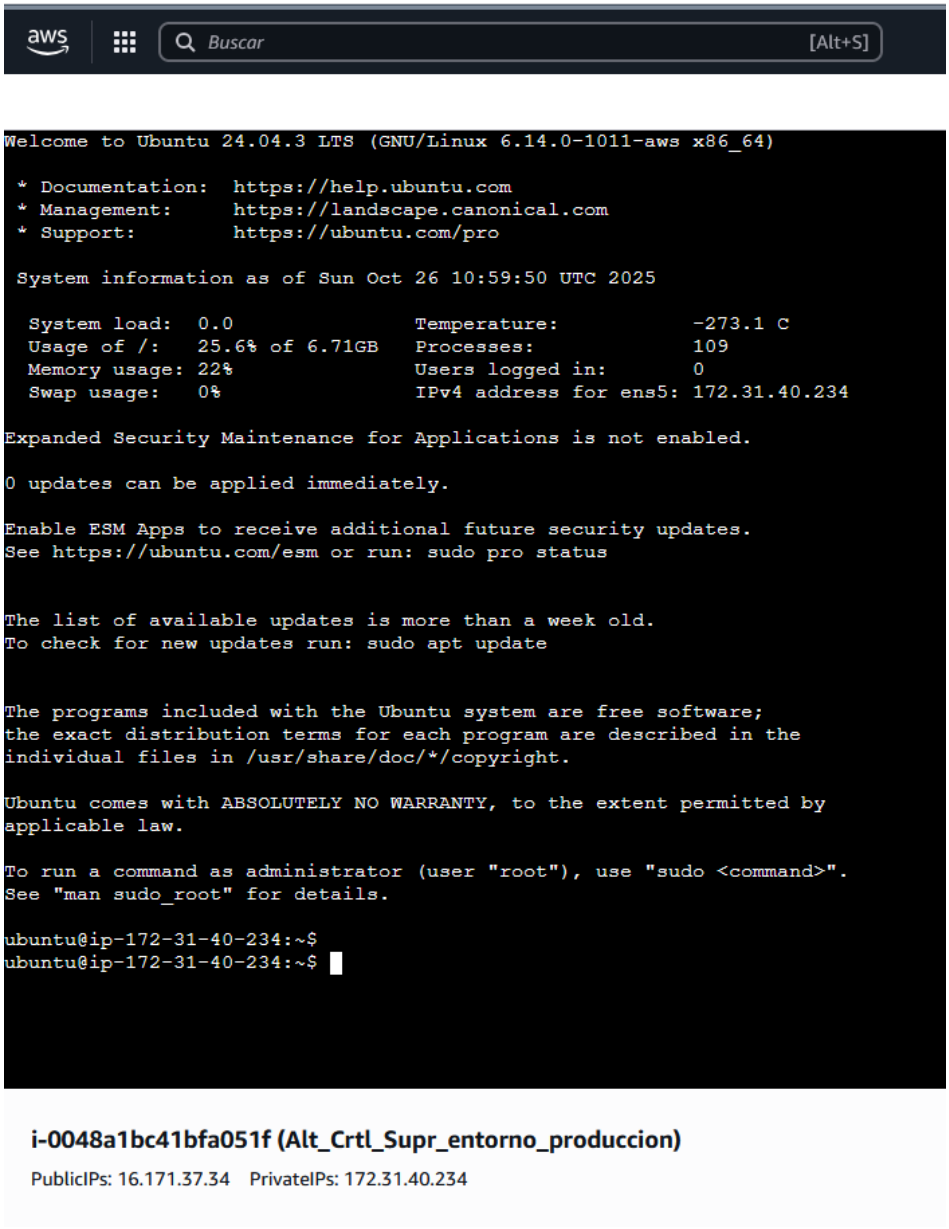


Y para ver que todo está correcto, lanzamos la instancia.

Deberíamos ver algo así para saber que se ha lanzado de forma correcta.

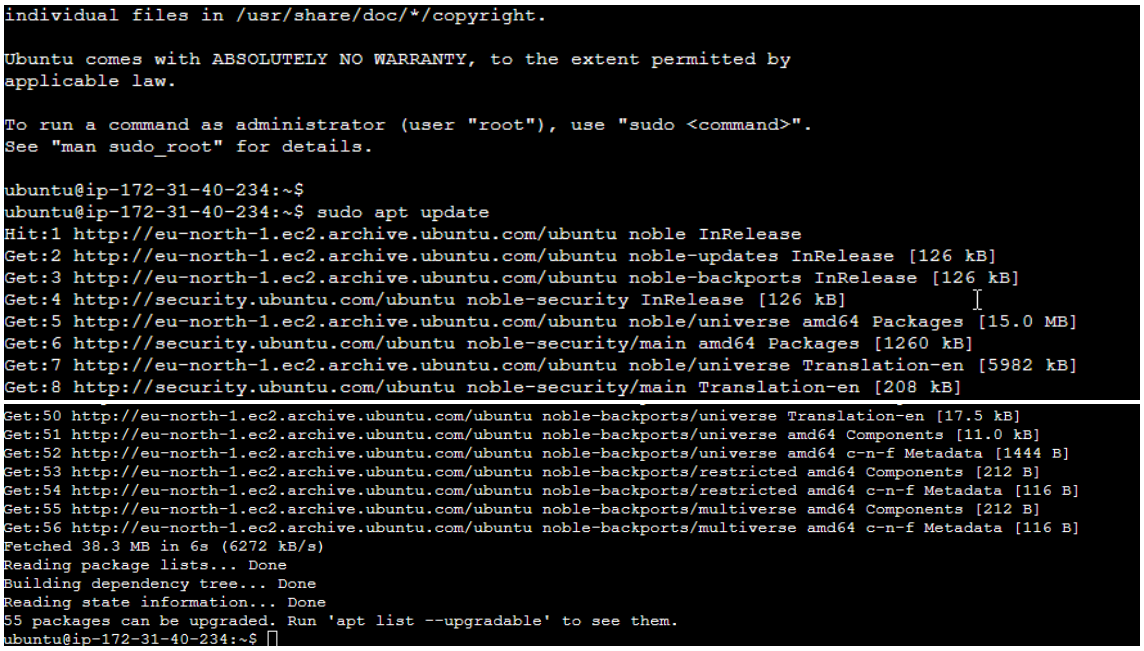


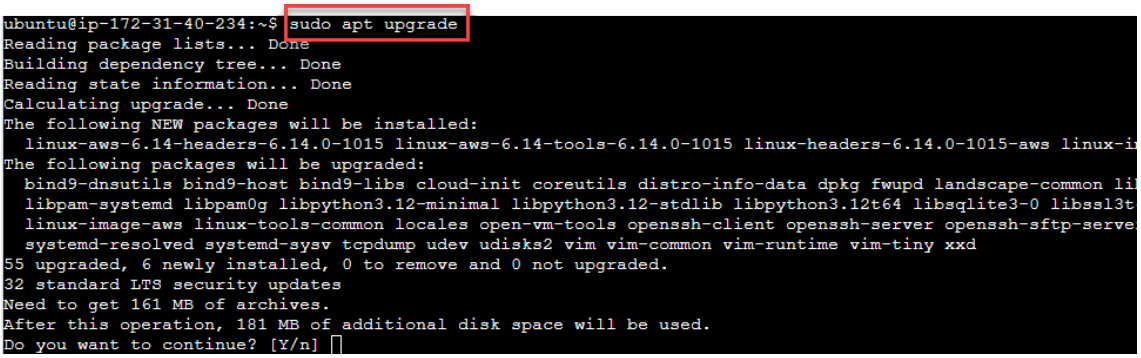
Aquí ya podríamos dar uso de nuestro sistema Ubuntu.



5. Instalar Jenkins.

Primero utilizaremos ***"sudo apt update"*** y ***"sudo apt upgrade"*** para preparar la terminal para el uso.





Jenkins requiere un entorno de ejecución de Java (JRE). La versión 17 es la mas recomendada. ***"sudo apt install openjdk-17-jre -y"*** para instalar Kenkins.

```
ubuntu@ip-172-31-40-234:~$ sudo apt install openjdk-17-jre -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-common at-spi2-core ca-certificates-java dconf-gsettings-backe
  fonts-dejavu-extra fonts-dejavu-mono gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-them
  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0t64 libatspi2.0-0t64 libavahi-client3 libavahi-common-data libavahi
  libdeflate0 libdrm-amdgpu1 libdrm-intel1 libfontconfig1 libgail-common libgail18t64 libgbm1 libgdk-pixbuf-2.0-0 libgdk-pix
  libglx-mesa0 libglx0 libgraphite2-3 libgtk2.0-0t64 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice6 libjbig0 libjpeg-t
  libpangoft2-1.0-0 libpciaccess0 libpcsclite1 libpixman-1-0 librsvg2-2 librsvg2-common libsharpyuv0 libsm6 libthai-data lib
  libx11-xcb1 libxaw7 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-randr0 libxcb-render0 libxcb-shape0 libxcb-shm0 libxc
  libxi6 libxinerama1 libxkbfile1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt6t64 libxtst6 libxv1 libxxf86dga
  session-migration ubuntu-mono x11-common x11-utils
Suggested packages:
  default-jre alsa-utils libasound2-plugins cups-common gvfs liblcms2-utils pcscd librsvg2-bin libnss-mdns fonts-ipafont-got
  mesa-utils
Recommended packages:
  luit
The following NEW packages will be installed:
  adwaita-icon-theme alsa-topology-conf alsa-ucm-conf at-spi2-common at-spi2-core ca-certificates-java dconf-gsettings-backe
  fonts-dejavu-extra fonts-dejavu-mono gsettings-desktop-schemas gtk-update-icon-cache hicolor-icon-theme humanity-icon-them
  libatk-wrapper-java libatk-wrapper-java-jni libatk1.0-0t64 libatspi2.0-0t64 libavahi-client3 libavahi-common-data libavahi
  libdeflate0 libdrm-amdgpu1 libdrm-intel1 libfontconfig1 libgail-common libgail18t64 libgbm1 libgdk-pixbuf-2.0-0 libgdk-pix
  libglx-mesa0 libglx0 libgraphite2-3 libgtk2.0-0t64 libgtk2.0-bin libgtk2.0-common libharfbuzz0b libice6 libjbig0 libjpeg-t
  libpangoft2-1.0-0 libpciaccess0 libpcsclite1 libpixman-1-0 librsvg2-2 librsvg2-common libsharpyuv0 libsm6 libthai-data lib
```

Con ***"java –version"*** sabremos la versión utilizada.

```
ubuntu@ip-172-31-40-234:~$ java --version
openjdk 17.0.16 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
ubuntu@ip-172-31-40-234:~$
```

Luego de instalar JAVA, debemos añadir el repositorio de Jenkins antes de instalarlo.

Ahora debemos añadir la clave de Jenkins para autenticar el repositorio y luego añadir la URL del repositorio a tus fuentes de apt.

***sudo wget -O /usr/share/keyrings/jenkins-keyring.asc ***
https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key

```
ubuntu@ip-172-31-40-234:~$ sudo wget -O /usr/share/keyrings/jenkins-keyring.asc \
  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
--2025-10-26 11:38:15--  https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key
Resolving pkg.jenkins.io (pkg.jenkins.io)... 199.232.174.133, 2a04:4e42:6b::645
Connecting to pkg.jenkins.io (pkg.jenkins.io)|199.232.174.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3175 (3.1K) [application/pgp-keys]
Saving to: '/usr/share/keyrings/jenkins-keyring.asc'

/usr/share/keyrings/jenkins-keyring.asc             100%[=======================================

2025-10-26 11:38:15 (46.3 MB/s) - '/usr/share/keyrings/jenkins-keyring.asc' saved [3175/3175]
```

Luego añadimos el repositorio con:

echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] https://pkg.jenkins.io/debian-stable binary/" | sudo tee /etc/apt/sources.list.d/jenkins.list > /dev/null

```
ubuntu@ip-172-31-40-234:~$ echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc]" \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null

ubuntu@ip-172-31-40-234:~$
```

Actualizamos con ***"sudo apt update"***

```
ubuntu@ip-172-31-40-234:~$ sudo apt update
Hit:1 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble InRelease
Hit:2 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-updates InRelease
Hit:3 http://eu-north-1.ec2.archive.ubuntu.com/ubuntu noble-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
Hit:7 http://security.ubuntu.com/ubuntu noble-security InRelease
Get:8 https://pkg.jenkins.io/debian-stable binary/ Packages [30.0 kB]
Fetched 32.9 kB in 0s (93.2 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
All packages are up to date.
ubuntu@ip-172-31-40-234:~$
```

Finalmente instalamos Jenkins con ***"sudo apt-get install jenkins"***

```
ubuntu@ip-172-31-40-234:~$ sudo apt-get install jenkins
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  net-tools
The following NEW packages will be installed:
  jenkins net-tools
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 95.2 MB of archives.
After this operation, 96.3 MB of additional disk space will be used.
Do you want to continue? [Y/n]
Fetched 95.2 MB in 2s (50.0 MB/s)
Selecting previously unselected package net-tools.
(Reading database ... 118075 files and directories currently installed.)
Preparing to unpack .../net-tools_2.10-0.1ubuntu4.4_amd64.deb ...
Unpacking net-tools (2.10-0.1ubuntu4.4) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.528.1_all.deb ...
Unpacking jenkins (2.528.1) ...
Setting up net-tools (2.10-0.1ubuntu4.4) ...
Setting up jenkins (2.528.1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/jenkins.service → /usr/lib/systemd/system/jenkins.service.
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Pending kernel upgrade!
Running kernel version:
  6.14.0-1011-aws
Diagnostics:
  The currently running kernel version is not the expected kernel version 6.14.0-1015-aws.

Restarting the system to load the new kernel will not be handled automatically, so you should consider rebooting.

Restarting services...

Service restarts being deferred:
 /etc/needrestart/restart.d/dbus.service
 systemctl restart getty@tty1.service
 systemctl restart networkd-dispatcher.service
 systemctl restart serial-getty@ttyS0.service
 systemctl restart systemd-logind.service
 systemctl restart unattended-upgrades.service

No containers need to be restarted.

User sessions running outdated binaries:
 ubuntu @ session #2: sshd[1088,1541]
 ubuntu @ user manager service: systemd[1435]

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-40-234:~$
```

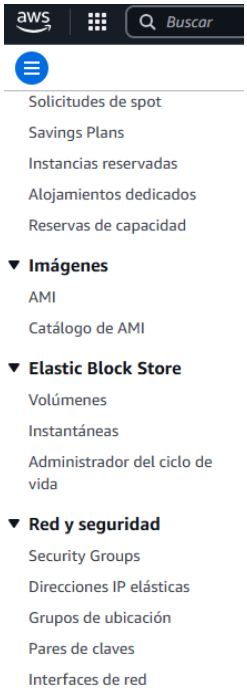
6. Configurar Jenkins para que despliegue automáticamente nuestros cambios en el código de la aplicación, publicados en GitHub.

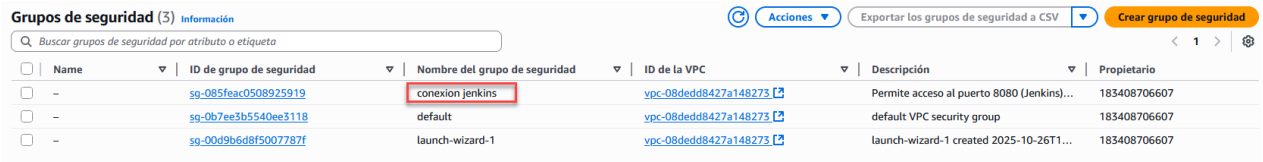
Habilitamos e iniciamos Jenkins con “sudo systemctl enable jenkins”
“sudo systemctl start jenkins”

```
ubuntu@ip-172-31-40-234:~$ sudo systemctl enable jenkins
Synchronizing state of jenkins.service with SysV service script with /usr/lib/systemd/systemd-sysv-install.
Executing: /usr/lib/systemd/systemd-sysv-install enable jenkins
ubuntu@ip-172-31-40-234:~$ sudo systemctl start jenkins
ubuntu@ip-172-31-40-234:~$ sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Sun 2025-10-26 11:54:52 UTC; 8min ago
   Main PID: 18486 (java)
      Tasks: 38 (limit: 1008)
     Memory: 294.0M (peak: 322.8M)
        CPU: 21.473s
     CGroup: /system.slice/jenkins.service
             └─18486 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPor

Oct 26 11:54:47 ip-172-31-40-234 jenkins[18486]: [LF]> This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Oct 26 11:54:47 ip-172-31-40-234 jenkins[18486]: [LF]>
Oct 26 11:54:47 ip-172-31-40-234 jenkins[18486]: [LF]> ************************************************************
Oct 26 11:54:47 ip-172-31-40-234 jenkins[18486]: [LF]> ************************************************************
Oct 26 11:54:47 ip-172-31-40-234 jenkins[18486]: [LF]> ************************************************************
Oct 26 11:54:52 ip-172-31-40-234 jenkins[18486]: 2025-10-26 11:54:52.813+0000 [id=32]        INFO        jenkins.InitReactorRunner$1#onA
Oct 26 11:54:52 ip-172-31-40-234 jenkins[18486]: 2025-10-26 11:54:52.844+0000 [id=23]        INFO        hudson.lifecycle.Lifecycle#onRe
Oct 26 11:54:52 ip-172-31-40-234 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Oct 26 11:54:53 ip-172-31-40-234 jenkins[18486]: 2025-10-26 11:54:53.627+0000 [id=49]        INFO        h.m.DownloadService$Downloadabl
Oct 26 11:54:53 ip-172-31-40-234 jenkins[18486]: 2025-10-26 11:54:53.628+0000 [id=49]        INFO        hudson.util.Retrier#start: Perf
lines 1-20/20 (END)
```

El siguiente paso será crear un grupo de seguridad en la consola de AWS, esto nos permitirá conectarnos con nuestra IP pública desde el navegador.

En el panel situado a la izquierda de la pantalla, bajamos hasta “Red y Seguridad” y clicamos en “Security Groups”



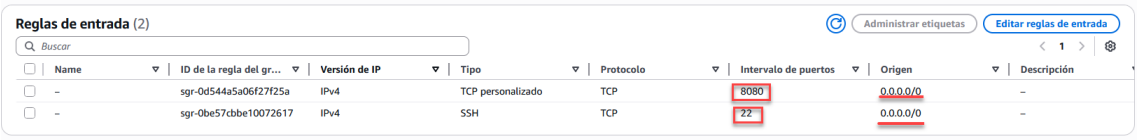


Creamos un nuevo grupo y editamos las reglas de entrada, agregando una nueva:

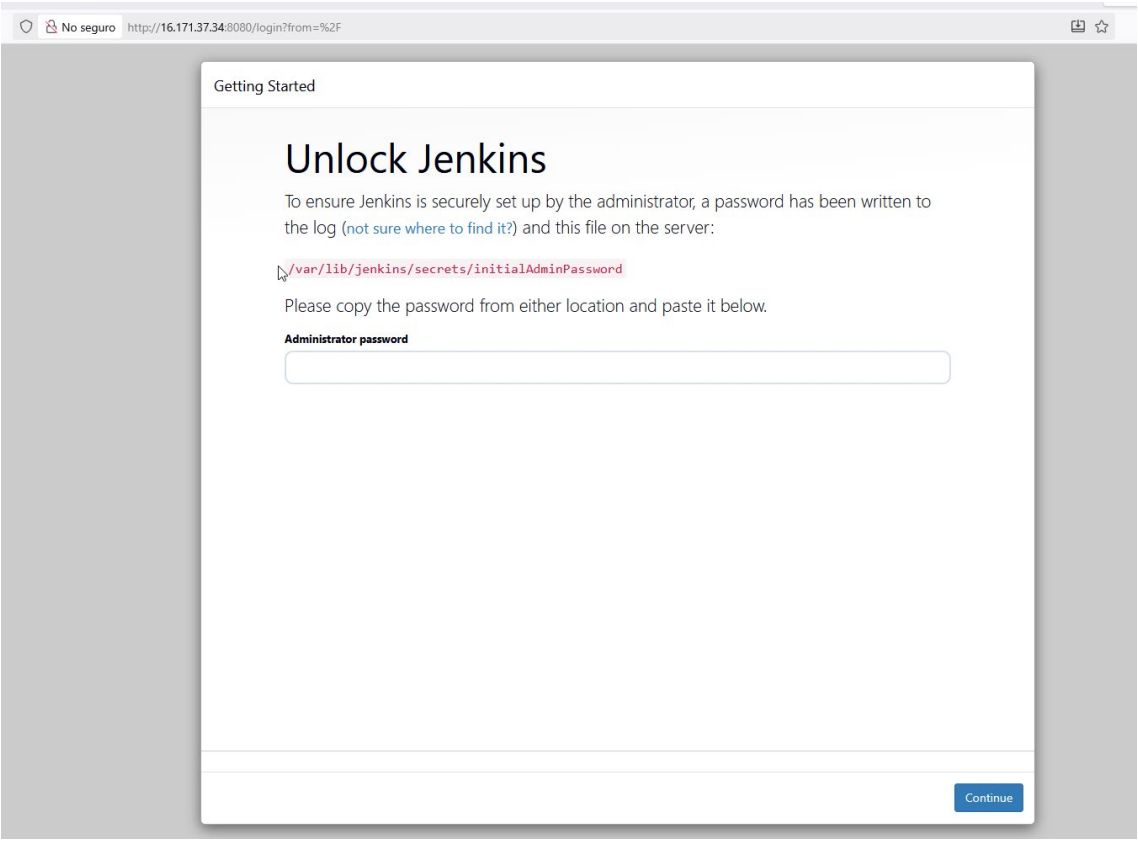
Tipo: TCP personalizado

Intervalo de puertos:8080

Origen: 0.0.0.0/0 si queremos que tenga acceso abierto.

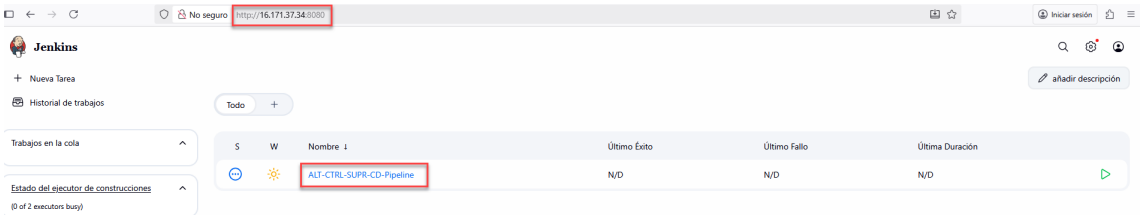


Una vez seguidos estos pasos, iremos a nuestro navegador y nos conectaremos a Jenkins http://16.171.37.34:8080/

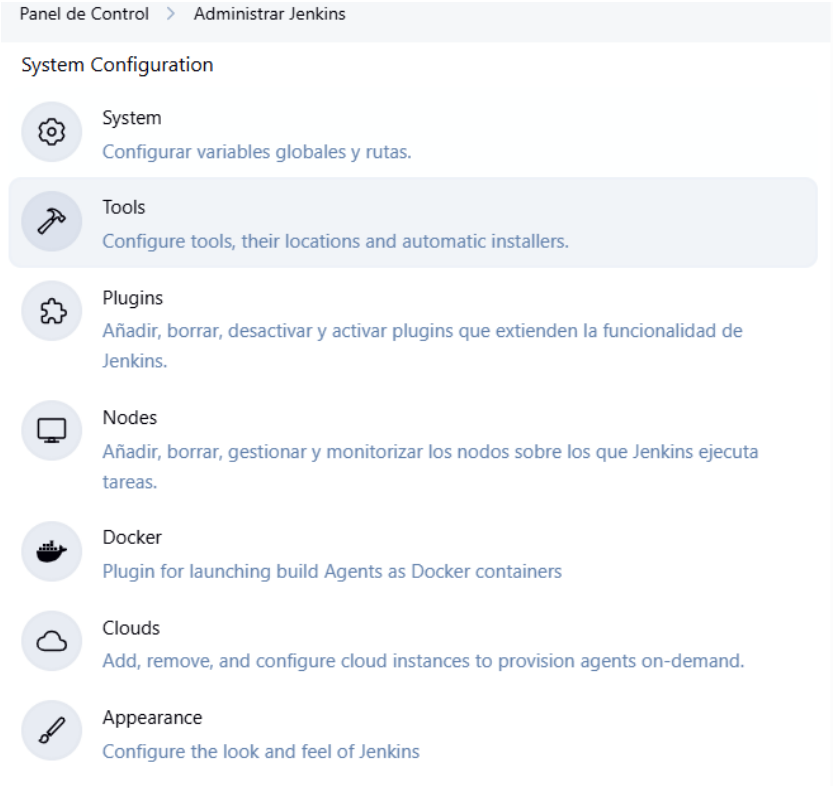


La contraseña de administrador por defecto la primera vez que nos conectemos la podemos conseguir con “sudo cat /var/lib/jenkins/secrets/initialAdminPassword”

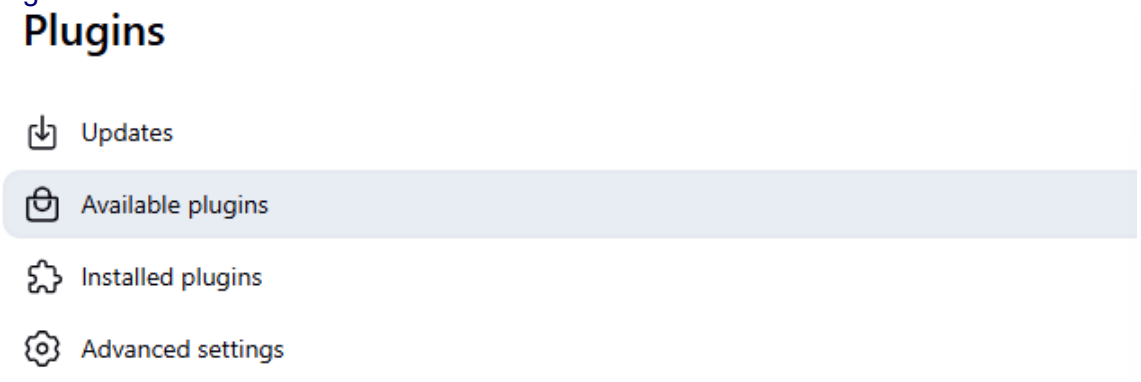
Una vez iniciada la sesión instalaremos los plugins necesarios para el despliegue automático cuando se realicen cambios en el código de la aplicación.



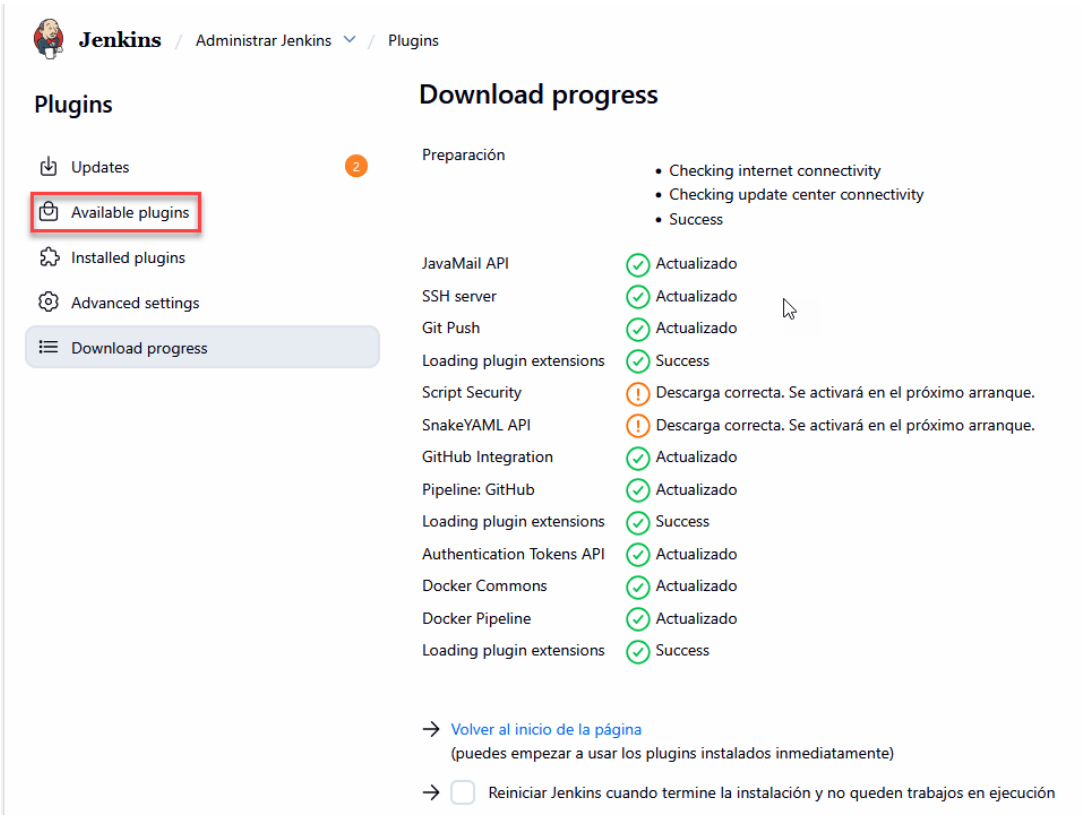
En el menú izquierdo - Administrar Jenkins y se desplegarán las opciones de configuración del sistema



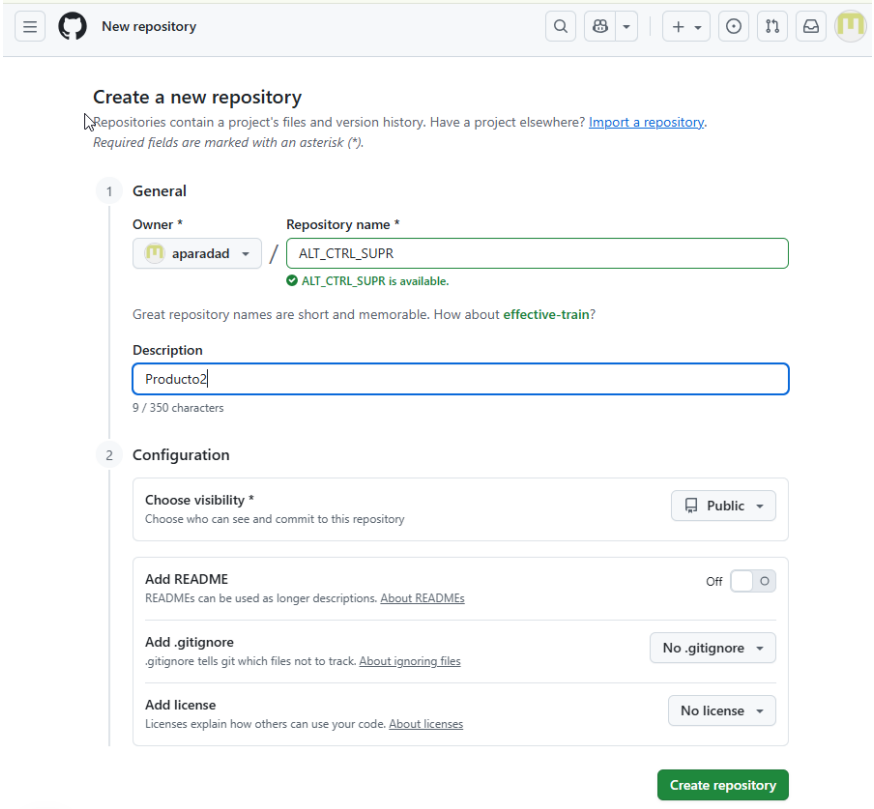
Iremos a Plugins y en la pestaña de “Available plugins” buscamos e instalamos las siguientes extensiones



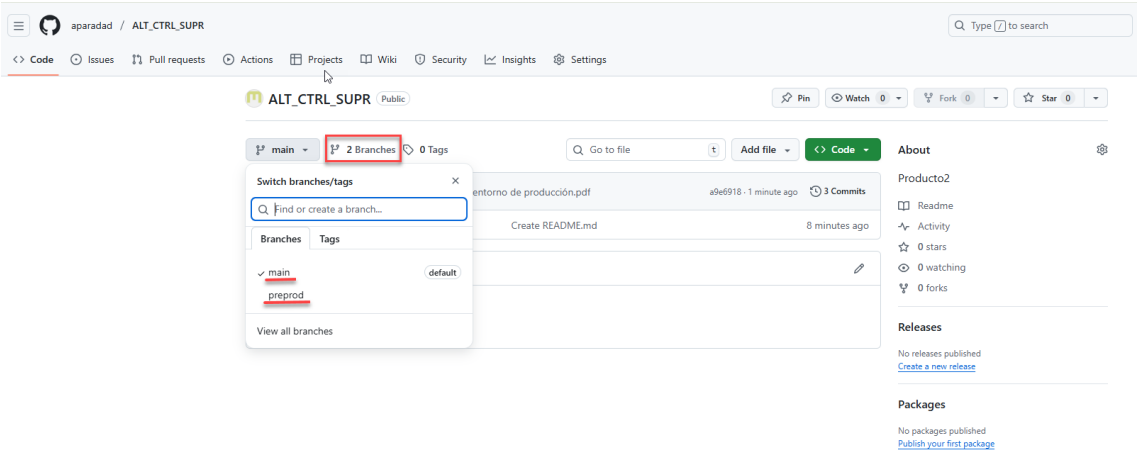
Git plugin, GitHub plugin, Pipeline, GitHub Integration Plugin y Docker Pipeline



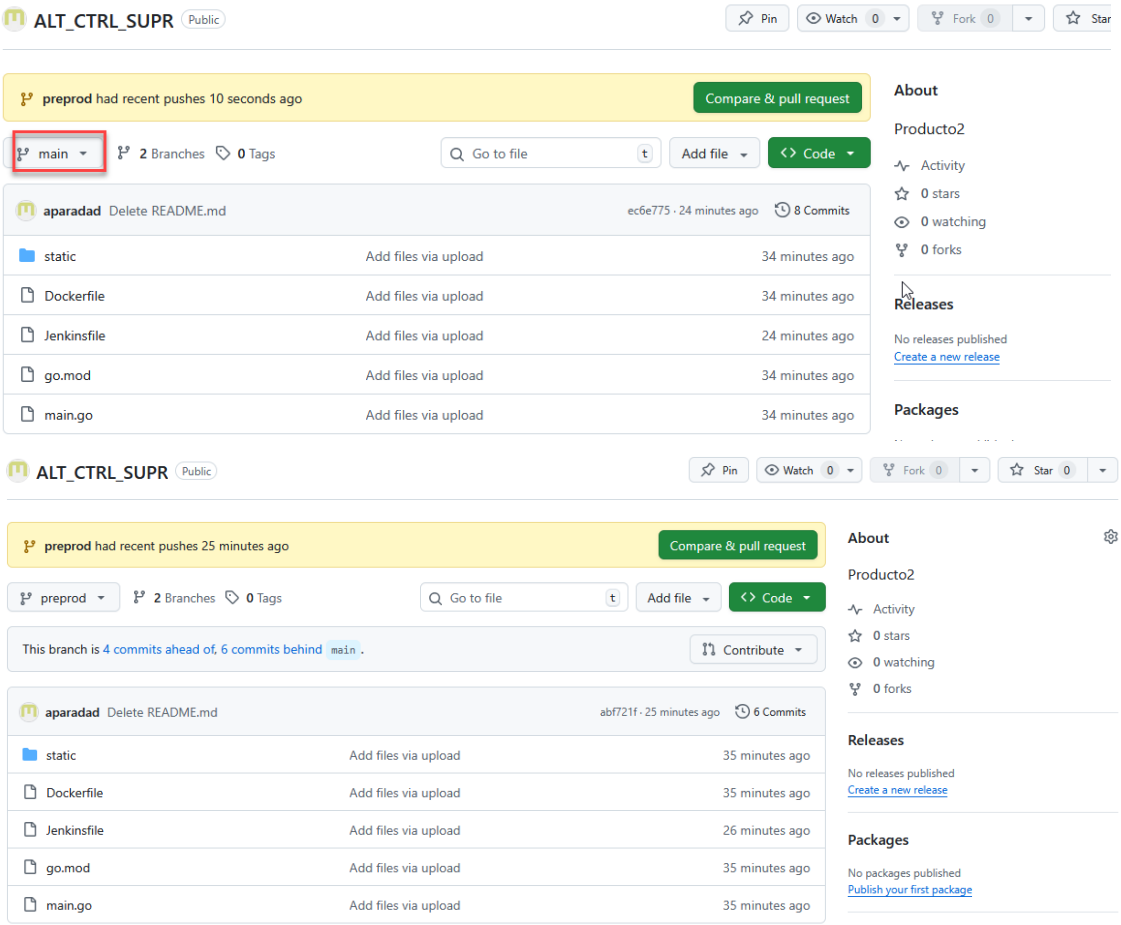
Si no teníamos anteriormente, creamos una cuenta en GitHub y un repositorio.



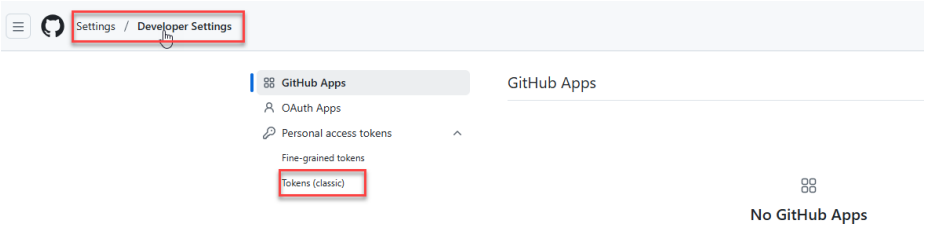
Creamos dos ramas la main que será la de producción y la preprod que es la de preproducción.



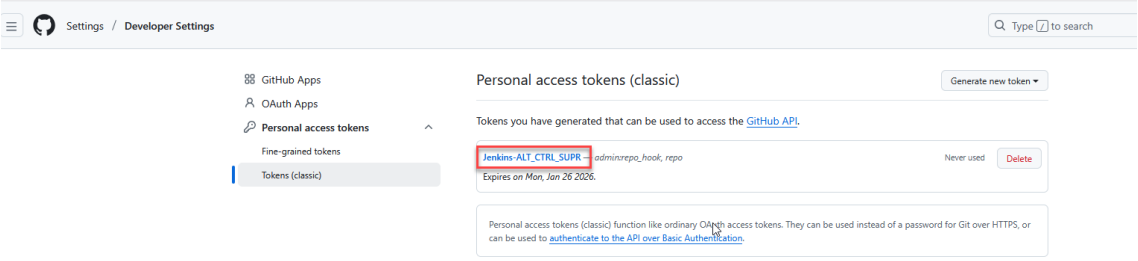
Subimos los ficheros necesarios a github a main y a preprod. Subimos nuestra aplicación de go (go.mod, main.go y la carpeta static con nuestra imagen), también subimos el dockerfile y el jenkinsfile que es el script que define el pipeline de Jenkins (Etapas Build, Push, Deploy.)



Para autorizar el acceso desde Jenkins iremos a la configuración de nuestra cuenta de GitHub y entramos en “developer settings” - “personal access tokens” - ”tokens”

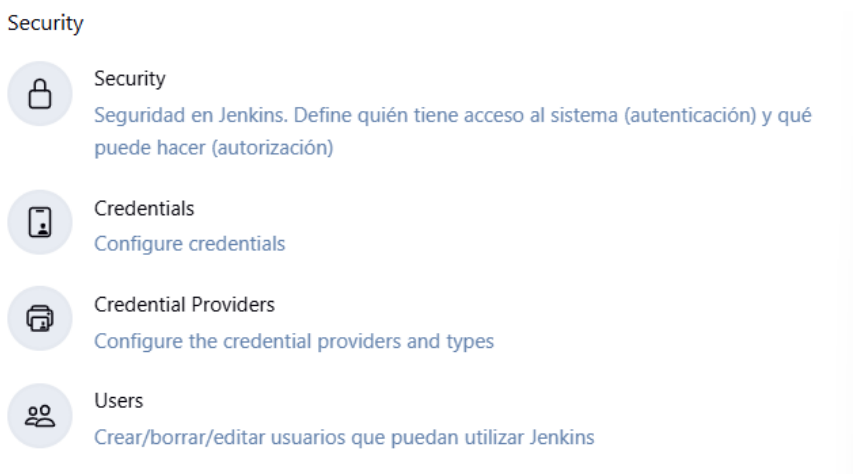


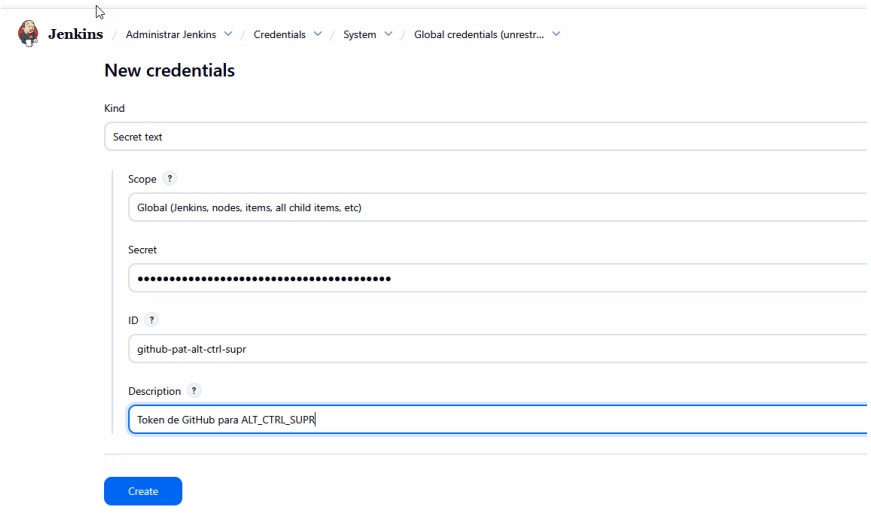
Generamos el token y lo guardamos en una carpeta segura y a la vista en nuestro PC



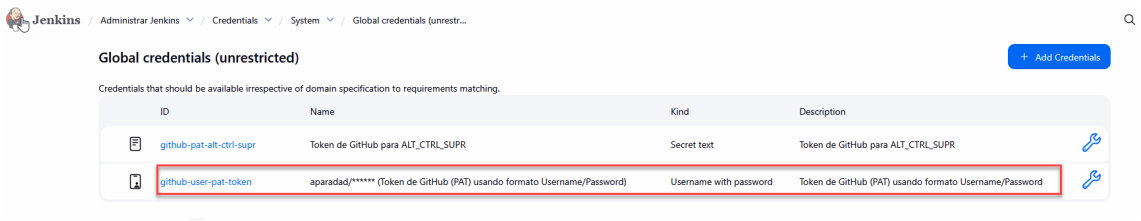

```
Token: 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
```

Volvemos a Jenkins para añadir nuestras credenciales de GitHub, panel de control - administrar jenkins y nos dirigimos al menú “Credentials”

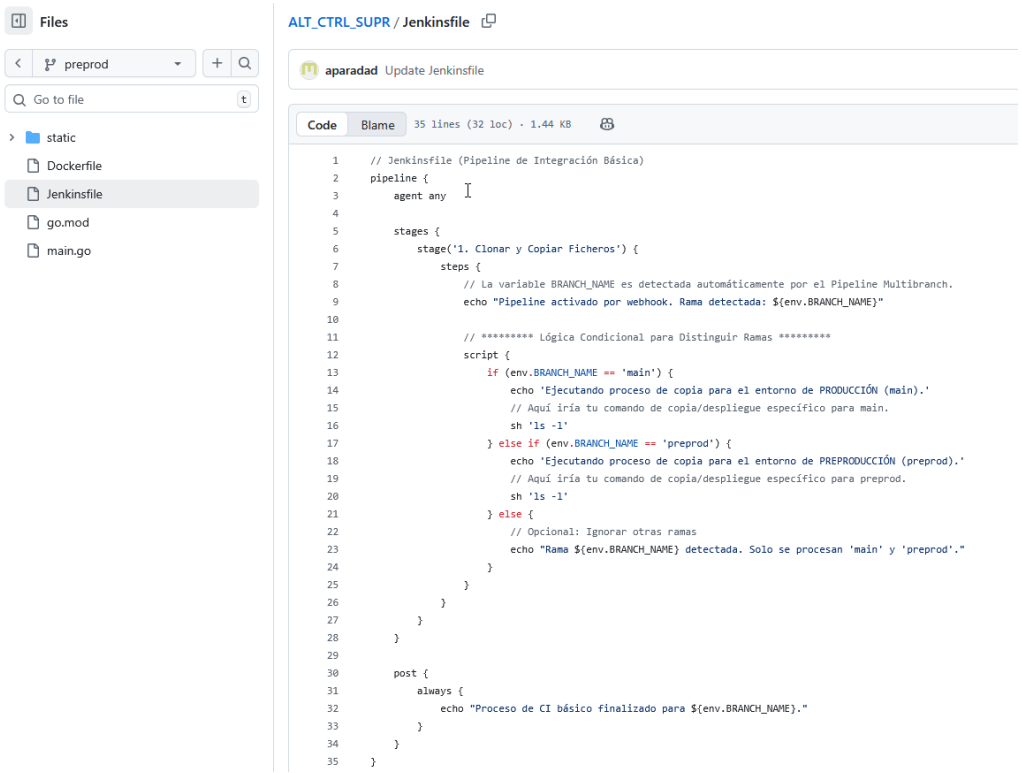




Donde añadiremos un tipo: usuario (usuario en GitHub) con contraseña (el token recientemente descargado)

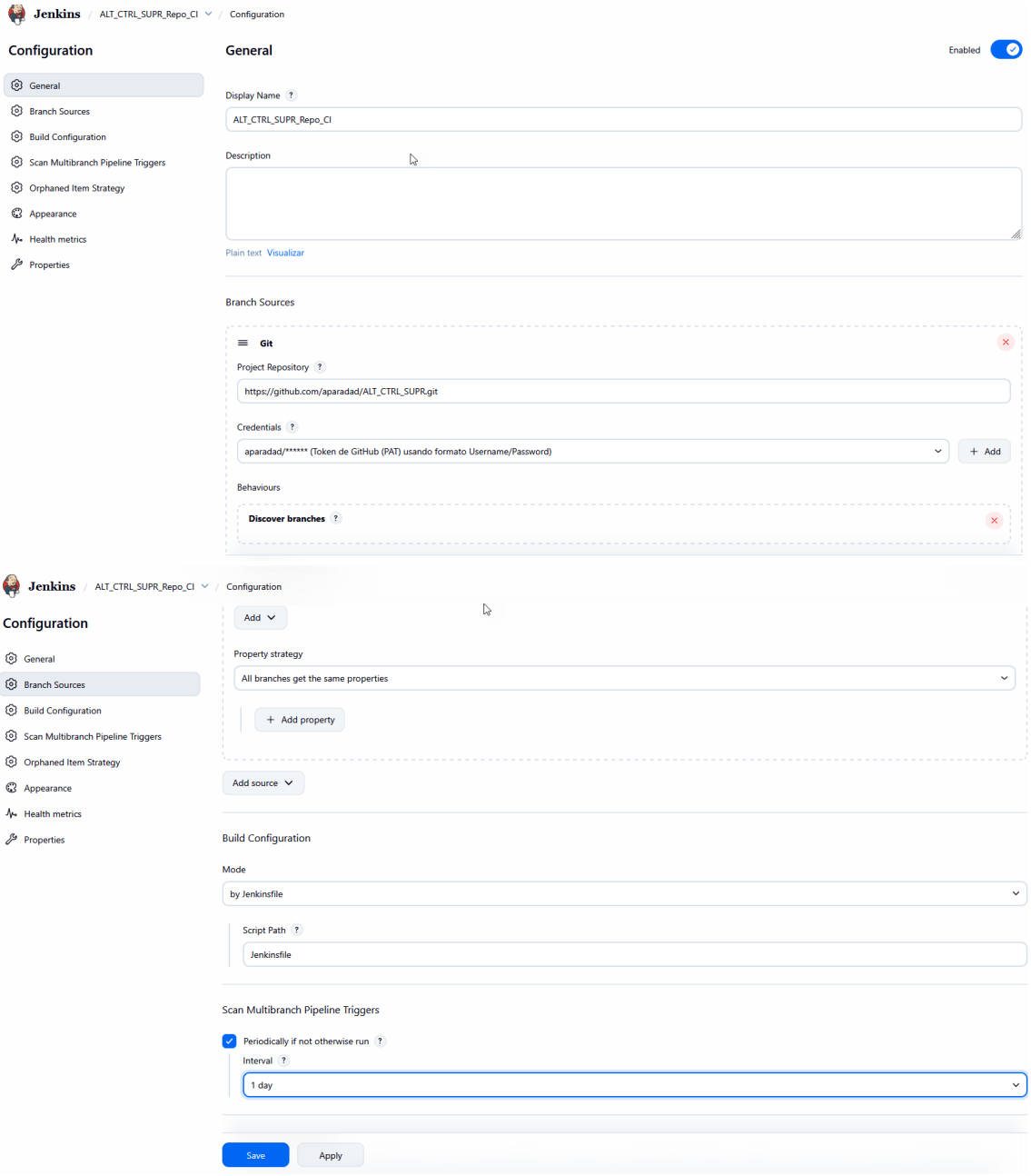


Creamos el fichero Jenkinsfile que solo confirma la recepción del código y lista los archivos en el *workspace*.

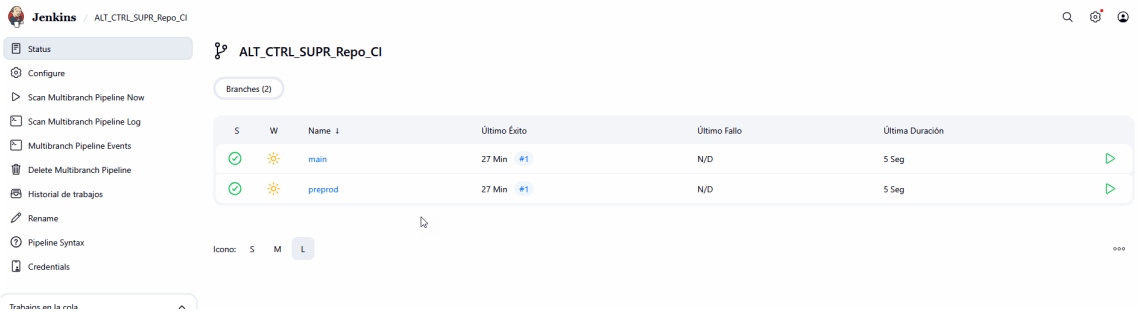


Como queremos manejar múltiples ramas automáticamente, creamos un **Pipeline Multibranch** en lugar de un *Pipeline* simple.

En el panel de control de Jenkins, seleccionamos nueva tarea. Ponemos el nombre ALT_CTRL_SUPR_Repo_CI.
Y seleccionamos **Multibranch Pipeline** (Pipeline Multibranch). Tenemos nuestro repositorio https://github.com/aparadad/ALT_CTRL_SUPR.git y seleccionamos nuestras credenciales que configuramos anteriormente.
En **Build Configuration** (Configuración de Construcción). Dejamos la opción **"Discover branches"** y marcamos **"Periodically if not otherwise run"** dejandolo en un dia (es útil como *fallback*).



Como vemos nos ha creado una tarea por cada rama. Ya que seleccionamos **Multibranch Pipeline** (Pipeline Multibranch).

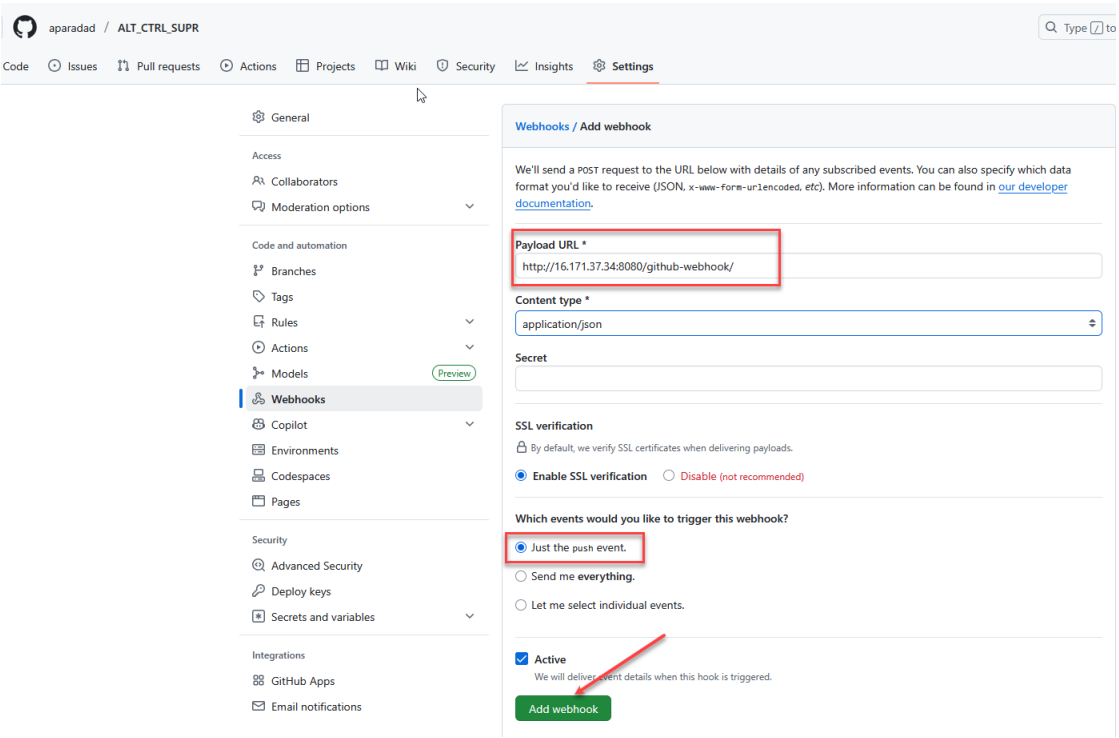


Configuración Final del Webhook en GitHub

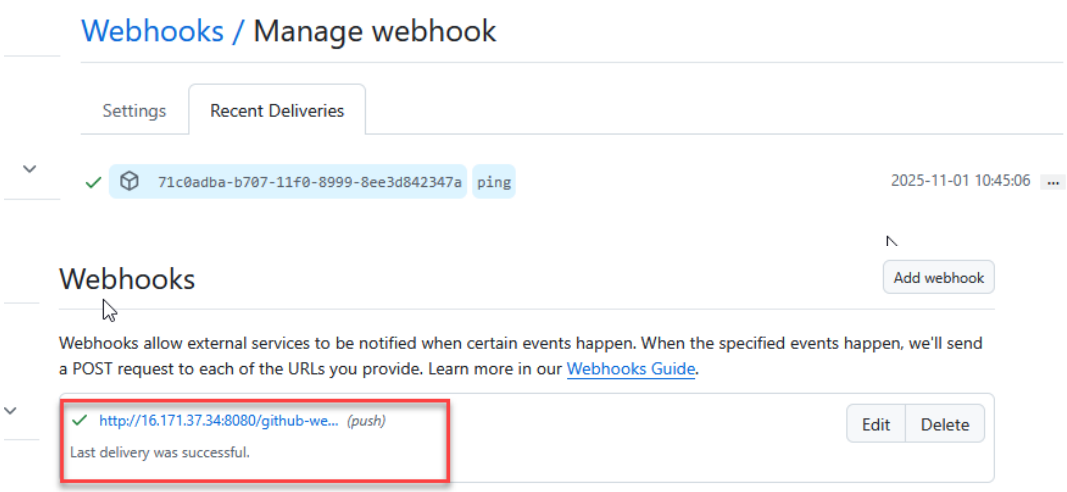
El URL del *webhook* que debes configurar en GitHub para que el *Pipeline* Multibranch se active es ligeramente diferente:

Vamos a GitHub (**Settings** > **Webhooks**). Y añadimos un Webhooks ponemos el **Payload URL** apuntando a la URL de escaneo de nuestro controlador http://16.171.37.34:8080/github-webhook/ y la opción **Eventos:** Solo eventos de push.

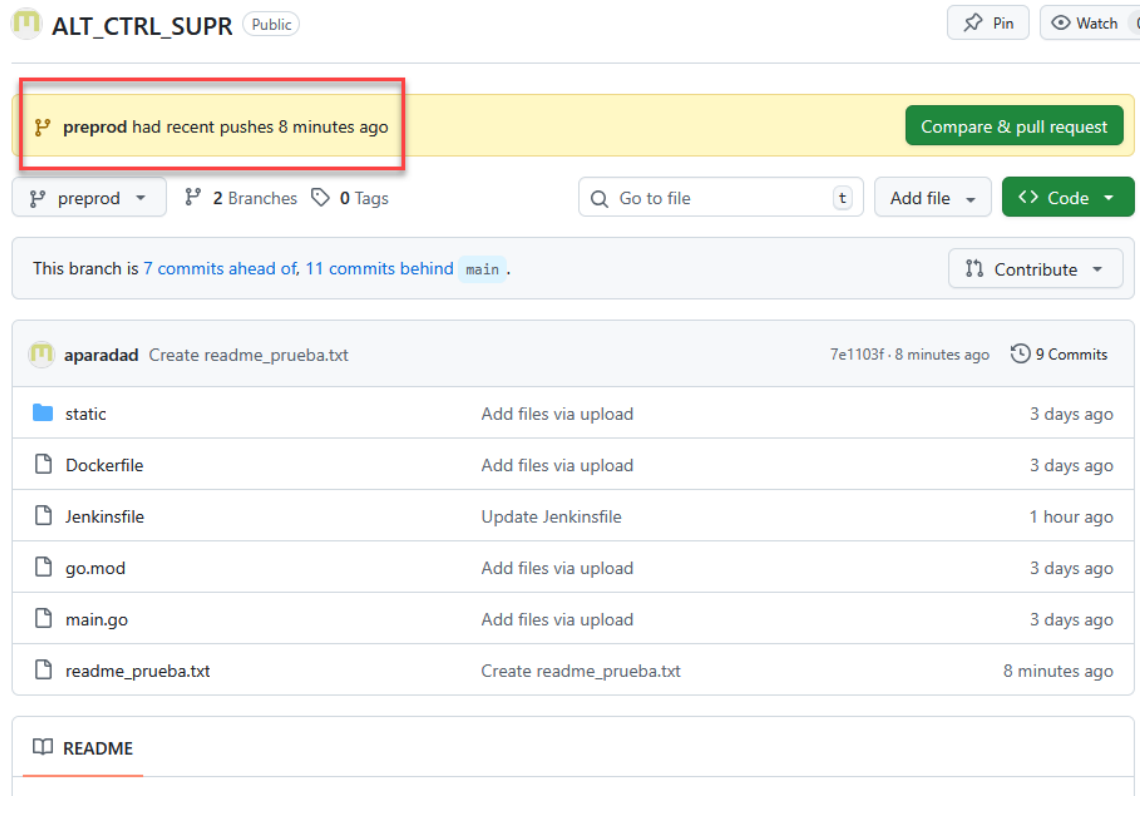
Con esta configuración (Multibranch Pipeline), Jenkins detectará que hay actividad en la rama main o preprod y ejecutará el Jenkinsfile de esa rama automáticamente.

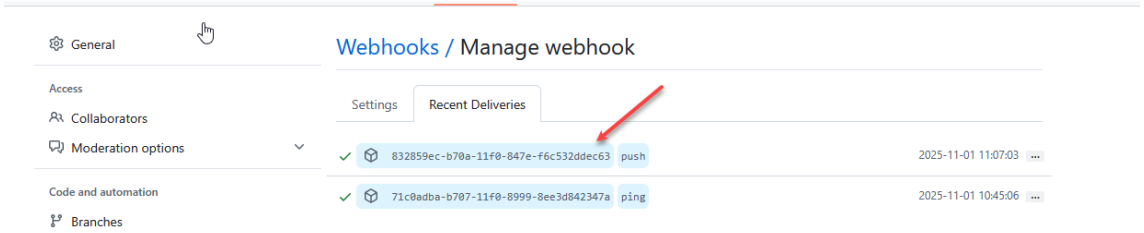


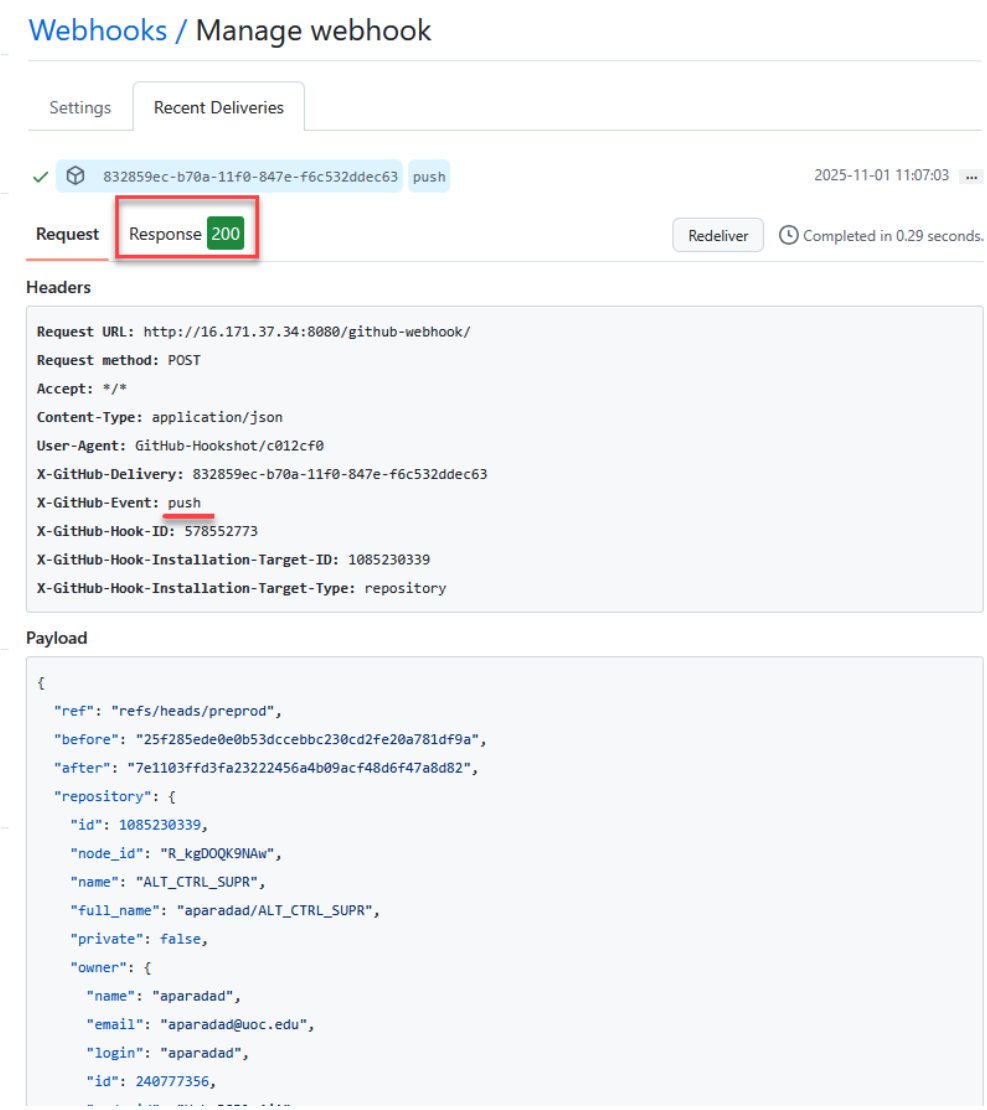
Una vez creado, GitHub enviará un pequeño "ping" de prueba a Jenkins.



Ahora vamos a realizar un cambio en nuestras ramas para ver que se realiza ese cambio automáticamente. Vamos a Github y creamos un fichero readme_prueba.txt en la rama preprod.

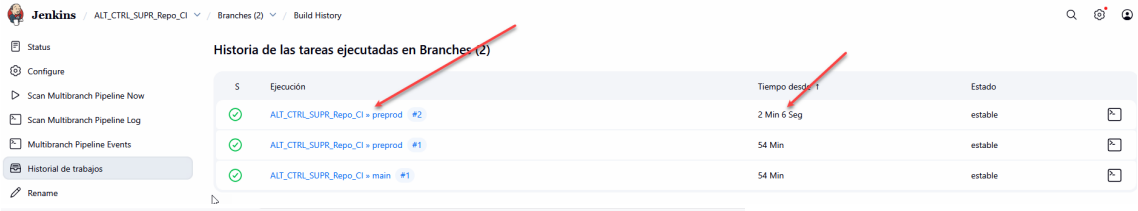


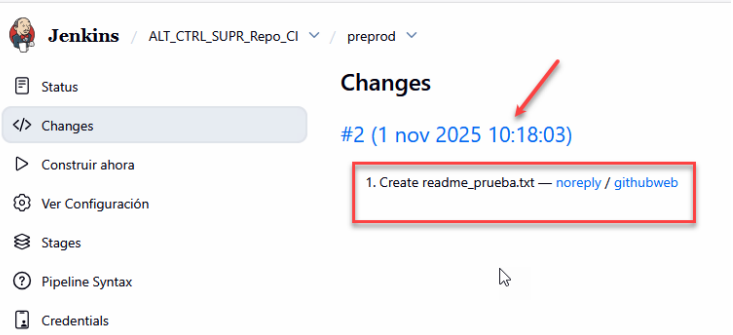



Webhooks / Manage webhook
Settings
Recent Deliveries
832859ec-b70a-11f0-847e-f6c532ddec63
push
2025-11-01 11:07:03
Request
Response 200
Redeliver
Completed in 0.29 seconds.
Headers
Request URL: http://16.171.37.34:8080/github-webhook/
Request method: POST
Accept: */*
Content-Type: application/json
User-Agent: GitHub-Hookshot/c012cf0
X-GitHub-Delivery: 832859ec-b70a-11f0-847e-f6c532ddec63
X-GitHub-Event: push
X-GitHub-Hook-ID: 578552773
X-GitHub-Hook-Installation-Target-ID: 1085230339
X-GitHub-Hook-Installation-Target-Type: repository
Payload
{
"ref": "refs/heads/preprod",
"before": "25f285ede0e0b53dccebbc230cd2fe20a781df9a",
"after": "7e1103ffd3fa23222456a4b09acf48d6f47a8d82",
"repository": {
"id": 1085230339,
"node_id": "R_kgDOQK9NAw",
"name": "ALT_CTRL_SUPR",
"full_name": "aparadad/ALT_CTRL_SUPR",
"private": false,
"owner": {
"name": "aparadad",
"email": "aparadad@uoc.edu",
"login": "aparadad",
"id": 240777356,

```
},
"created": false,
"deleted": false,
"forced": false,
"base_ref": null,
"compare": "https://github.com/aparadad/ALT_CTRL_SUPR/compare/25f285ede0e0...7e1103ffd3fa",
"commits": [
  {
    "id": "7e1103ffd3fa23222456a4b09acf48d6f47a8d82",
    "tree_id": "3b68d963ed2bc165cd35f9ffb22843690257b83a",
    "distinct": true,
    "message": "Create readme_prueba.txt",
    "timestamp": "2025-11-01T11:07:02+01:00",
    "url": "https://github.com/aparadad/ALT_CTRL_SUPR/commit/7e1103ffd3fa23222456a4b09acf48d6f47a8d82",
    "author": {
      "name": "aparadad",
      "email": "aparadad@uoc.edu",
      "username": "aparadad"
    },
    "committer": {
      "name": "GitHub",
      "email": "noreply@github.com",
      "username": "web-flow"
    },
    "added": [
      "readme_prueba.txt"
    ],
    "removed": [

    ],
    "modified": [

    ]
  }
],
"head_commit": {
  "id": "7e1103ffd3fa23222456a4b09acf48d6f47a8d82",
  "tree_id": "3b68d963ed2bc165cd35f9ffb22843690257b83a",
  "distinct": true,
  "message": "Create readme_prueba.txt",
  "timestamp": "2025-11-01T11:07:02+01:00",
```


En jenkins lo podemos ver en el Historial de trabajos, pinchamos en el último, en cambios y vemos:





En la maquina Ubuntu lo podemos ver en nuestro workspace.

```
ubuntu@ip-172-31-40-234:/var/lib/jenkins/workspace/ALT_CTRL_SUPR_Repo_CI_preprod$ ls
Dockerfile  Jenkinsfile  go.mod  main.go  readme_prueba.txt  static
ubuntu@ip-172-31-40-234:/var/lib/jenkins/workspace/ALT_CTRL_SUPR_Repo_CI_preprod$ cd /var/lib/jenkins/workspace/ALT-CTRL
-SUPR-CD-Pipeline
ubuntu@ip-172-31-40-234:/var/lib/jenkins/workspace/ALT-CTRL-SUPR-CD-Pipeline$ ls
Dockerfile  Jenkinsfile  go.mod  main.go  static
ubuntu@ip-172-31-40-234:/var/lib/jenkins/workspace/ALT-CTRL-SUPR-CD-Pipeline$
```

7. Instalar MiniKube en nuestra máquina virtual.

Seguimos las indicaciones de la página oficial

1 Installation

Click on the buttons that describe your target platform. For other architectures, see the release page for a complete list of minikube binaries.

Operating system: **Linux** | macOS | Windows

Architecture: **x86-64** | ARM64 | ARMv7 | ppc64 | S390x

Release type: **Stable**

Installer type: **Binary download** | Debian package | RPM package

To install the latest minikube **stable** release on **x86-64 Linux** using **binary download**:

```
curl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
sudo install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
```

Copy

```
root@ip-172-31-39-237:/home/ubuntu# curl -LO https://github.com/kubernetes/minikube/releases/latest/download/minikube-linux-amd64
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  133M  100  133M    0     0   138M      0 --:--:-- --:--:-- --:--:--  138M
root@ip-172-31-39-237:/home/ubuntu# ls
minikube-linux-amd64
root@ip-172-31-39-237:/home/ubuntu# install minikube-linux-amd64 /usr/local/bin/minikube && rm minikube-linux-amd64
```

```
root@ip-172-31-39-237:/home/ubuntu# minikube version
minikube version: v1.37.0
commit: 65318f4cfff9c12cc87ec9eb8f4cdd57b25047f3
```

Al ejecutar **minikube start** ,nos hemos topado con un error que no está instalado docker.

Procedemos a instalarlo

```
ubuntu@ip-172-31-39-237:~$ sudo apt install -y curl apt-transport-https docker.io conntrack
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
curl is already the newest version (8.5.0-2ubuntu10.6).
curl set to manually installed.
The following packages were automatically installed and are no longer required:
  linux-aws-6.14-headers-6.14.0-1011 linux-aws-6.14-tools-6.14.0-1011 linux-headers-6.14.0-1011-aws linux-image-6.14.0-1011-aws linux-modules-6.14.0-1011-aws
  linux-tools-6.14.0-1011-aws
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
```

```
ubuntu@ip-172-31-39-237:~$ minikube start --driver=docker
* minikube v1.37.0 on Ubuntu 24.04
* Using the docker driver based on user configuration

X Exiting due to PROVIDER_DOCKER_NEWGRP: "docker version --format <no value>:<no value>:<no value>" exit status 1: permission denied while trying to connect to the Doc
ker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.50/version": dial unix /var/run/docker.sock: connect: permission denied
* Suggestion: Add your user to the 'docker' group: 'sudo usermod -aG docker $USER && newgrp docker'
* Documentation: https://docs.docker.com/engine/install/linux-postinstall/
```

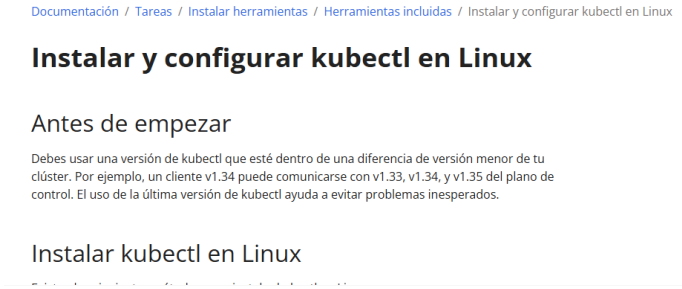
```
ubuntu@ip-172-31-39-237:~$ minikube start --driver=docker
* minikube v1.37.0 on Ubuntu 24.04
* Using the docker driver based on user configuration

X Exiting due to RSRC_INSUFFICIENT_CONTAINER_MEMORY: docker only has 914MiB available, less than the required 1800MiB for Kubernetes
```

Hemos cambiado la instancia de t3.micro a t3.medium. Sigue dandonos error así que hemos ampliado el volumen de 8GB a 20GB

Utilizamos **minikube start –driver=docker –memory=2048 –cpu=2 –preload=false** para restringir el uso de memory y de cpus que usará minikube.

```
Last login: Fri Nov  7 11:42:41 2025 from 3.120.181.44
ubuntu@ip-172-31-47-108:~$ df -h /
Filesystem      Size  Used Avail Use% Mounted on
/dev/root        19G  5.2G   14G  29% /
ubuntu@ip-172-31-47-108:~$ minikube start --driver=docker --memory=2048 --cpus=2 --preload=false
* minikube v1.37.0 on Ubuntu 24.04
* Using the docker driver based on user configuration
* Using Docker driver with root privileges
* Starting "minikube" primary control-plane node in "minikube" cluster
* Pulling base image v0.0.48 ...
* Creating docker container (CPUs=2, Memory=2048MB) ...
    > kubectl.sha256:  64 B / 64 B [-------------------------] 100.00% ? p/s 0s
    > kubeadm.sha256:  64 B / 64 B [-------------------------] 100.00% ? p/s 0s
    > kubelet.sha256:  64 B / 64 B [-------------------------] 100.00% ? p/s 0s
    > kubectl:  57.75 MiB / 57.75 MiB [----------] 100.00% 252.01 MiB p/s 400ms
    > kubelet:  56.45 MiB / 56.45 MiB [----------] 100.00% 208.42 MiB p/s 500ms
    > kubeadm:  70.60 MiB / 70.60 MiB [----------] 100.00% 141.70 MiB p/s 700ms
* Preparing Kubernetes v1.34.0 on Docker 28.4.0 ...
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
ubuntu@ip-172-31-47-108:~$
```

8. Instalar Kubernetes en nuestra máquina virtual.



Documentación / Tareas / Instalar herramientas / Herramientas incluidas / Instalar y configurar kubectl en Linux

Instalar y configurar kubectl en Linux

Antes de empezar

Debes usar una versión de kubectl que esté dentro de una diferencia de versión menor de tu clúster. Por ejemplo, un cliente v1.34 puede comunicarse con v1.33, v1.34, y v1.35 del plano de control. El uso de la última versión de kubectl ayuda a evitar problemas inesperados.

Instalar kubectl en Linux

Seguimos las instrucciones de la página oficial.

curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"

Este comando descarga la versión estable más reciente de kubectl directamente desde el repositorio oficial de Kubernetes.

```
ubuntu@ip-172-31-47-108:~$ curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   138  100   138    0     0    895      0 --:--:-- --:--:-- --:--:--   896
100 57.7M  100 57.7M    0     0   155M      0 --:--:-- --:--:-- --:--:--  155M
ubuntu@ip-172-31-47-108:~$
```

Una vez descargado el archivo, es importante **comprobar que no ha sido modificado o dañado durante la descarga**.

Para ello, realizamos los siguientes pasos:

- Descargar el archivo de comprobación de kubectl:
  curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl.sh 256"
- Validar el binario kubectl con el archivo de comprobación:
  echo "$(cat kubectl.sha256) kubectl" | sha256sum --check

  Si es válido, la salida será “Kubectl: OK” como vemos en la siguiente captura.

```
ubuntu@ip-172-31-47-108:~$ echo "$(cat kubectl.sha256)  kubectl" | sha256sum --check
kubectl: OK
ubuntu@ip-172-31-47-108:~$
```

Una vez verificada la integridad del archivo, procedemos a instalar kubectl en el sistema.

Para ello, ejecutamos el siguiente comando:
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl

Una vez completada la instalación, verificamos que la herramienta kubectl se ha instalado correctamente.

Para ello, ejecutamos el siguiente comando en la terminal:
kubectl version --client

El resultado muestra la versión del cliente de Kubernetes instalada en la máquina.

```
ubuntu@ip-172-31-47-108:~$ kubectl version --client
Client Version: v1.34.1
Kustomize Version: v5.7.1
ubuntu@ip-172-31-47-108:~$
```

9. Configurar Kubernetes para que nuestro contenedor se encuentre en alta disponibilidad (balanceo de carga entre mínimo dos instancias).

En este paso configuramos Kubernetes para que nuestro contenedor se ejecute con **alta disponibilidad**, garantizando que el servicio siga activo incluso si alguna instancia falla.

Esto se logra mediante la creación de un **deployment** con varias réplicas y la exposición del servicio a través de un **LoadBalancer**.

Primero, la imagen Docker de la aplicación utilizando el Dockerfile del proyecto y la etiqueta como **altctrlsupr/app:dev**.

docker build -t altctrlsupr/app:dev .

```
ubuntu@ip-172-31-47-108:~/Producto2$ ls
Dockerfile  k8s.yaml  main.go  static
ubuntu@ip-172-31-47-108:~/Producto2$ docker build -t altctrlsupr/app:dev .
```

El listado confirma que la imagen **altctrlsupr/app:dev** está almacenada dentro del Docker de Minikube.

```
ubuntu@ip-172-31-47-108:~/Producto2$ docker images
REPOSITORY                TAG           IMAGE ID       CREATED              SIZE
altctrlsupr/app           dev           d11322f8cc3d   About a minute ago   7.03MB
```

A continuación, creamos el archivo k8s.yaml, contiene la configuración necesaria para desplegar nuestra aplicación en Kubernetes. Definimos un **Deployment** con varias réplicas del contenedor y un **Service** tipo LoadBalancer que permite acceder a la aplicación desde fuera del clúster. (anexo 1)

kubectl apply -f k8s.yaml

```
ubuntu@ip-172-31-47-108:~/Producto2$ kubectl apply -f k8s.yaml
deployment.apps/uoc created
service/uoc created
```

Este comando crea tanto el Deployment como el Service definidos en el
archivo.

La salida confirma que Kubernetes ha creado correctamente los recursos:

- deployment.apps/uoc created
- service/uoc created

Finalmente, verificamos que las tres réplicas del contenedor están funcionando correctamente con el siguiente comando:

kubectl get pods

```
ubuntu@ip-172-31-47-108:~/Producto2$ kubectl get pods
NAME                   READY   STATUS    RESTARTS   AGE
uoc-74c8985855-9vrnx   1/1     Running   0          24s
uoc-74c8985855-fsmsd   1/1     Running   0          24s
uoc-74c8985855-msktz   1/1     Running   0          24s
```

El resultado muestra tres pods en estado Running, confirmando que el balanceo de carga está activo y que el servicio se encuentra en **alta disponibilidad.**

10. Comprobar que el balanceo funciona correctamente.

Una vez configurado el *deployment* con tres réplicas, comprobamos que el **balanceador de carga** funciona correctamente, distribuyendo las peticiones entre los diferentes pods del clúster.

Para ello, primero iniciamos un túnel en **Minikube** que permite acceder a los servicios tipo *LoadBalancer* desde el exterior:

```
ubuntu@ip-172-31-47-108:~$ minikube tunnel
Status:
        machine: minikube
        pid: 8771
        route: 10.96.0.0/12 -> 192.168.49.2
        minikube: Running
        services: [uoc]
    errors:
                minikube: no errors
                router: no errors
                loadbalancer emulator: no errors
```

El comando crea una ruta entre la red interna de Minikube y la máquina anfitriona.
En la salida podemos observar que el servicio uoc está activo y sin errores.

A continuación, listamos los servicios disponibles con el siguiente comando:

kubectl get services

El resultado muestra el servicio uoc con una dirección IP externa asignada por el balanceador de carga.

```
ubuntu@ip-172-31-47-108:~/Producto2$ kubectl get services
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)        AGE
kubernetes   ClusterIP      10.96.0.1        <none>           443/TCP        7d22h
uoc          LoadBalancer   10.111.131.241   10.111.131.241   80:32279/TCP   2m8s
```

Para verificar que las peticiones se reparten correctamente entre los pods, realizamos múltiples solicitudes HTTP al servicio utilizando un bucle:

for i in {1..10}; do curl http://10.111.131.241; done

Cada solicitud devuelve la página creada en el Producto 1.

```
ubuntu@ip-172-31-47-108:~/Producto2$ for i in {1..10}; do curl http://10.111.131.241; done
<!doctype html>
<html lang="es">
<head><meta charset="utf-8"><title>UOC Go Web</title>
<style>
  body { font-family:sans-serif; margin:2rem }
  img { max-width:360px; height:auto; border-radius:12px }
</style>
</head>
<body>
  <h1>Soy alumno de la UOC</h1>
  <img src="/static/uoc.jpg" alt="Imagen UOC">
</body></html><!doctype html>
<html lang="es">
<head><meta charset="utf-8"><title>UOC Go Web</title>
<style>
  body { font-family:sans-serif; margin:2rem }
  img { max-width:360px; height:auto; border-radius:12px }
</style>
```

Finalmente, verificamos en los **logs de cada pod** cómo se distribuyen las peticiones de forma equilibrada:

kubectl logs uoc-54d7b5b8f-lsvck --follow
kubectl logs uoc-54d7b5b8f-w241k --follow
kubectl logs uoc-54d7b5b8f-wgf8x --follow

En los registros se puede observar que las peticiones GET se reparten entre los tres pods en diferentes momentos:

kubectl logs uoc-54d7b5b8f-lsvck --follow

```
ubuntu@ip-172-31-47-108:~$ kubectl logs uoc-54d7b5b8f-lsvck --follow
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/11/07 12:21:06 [notice] 1#1: using the "epoll" event method
2025/11/07 12:21:06 [notice] 1#1: nginx/1.29.3
2025/11/07 12:21:06 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2025/11/07 12:21:06 [notice] 1#1: OS: Linux 6.14.0-1016-aws
2025/11/07 12:21:06 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/11/07 12:21:06 [notice] 1#1: start worker processes
2025/11/07 12:21:06 [notice] 1#1: start worker process 28
2025/11/07 12:21:06 [notice] 1#1: start worker process 29
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
```

kubectl logs uoc-54d7b5b8f-w24lk --follow

```
ubuntu@ip-172-31-47-108:~$ kubectl logs uoc-54d7b5b8f-w24lk --follow
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/11/07 12:21:13 [notice] 1#1: using the "epoll" event method
2025/11/07 12:21:13 [notice] 1#1: nginx/1.29.3
2025/11/07 12:21:13 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2025/11/07 12:21:13 [notice] 1#1: OS: Linux 6.14.0-1016-aws
2025/11/07 12:21:13 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/11/07 12:21:13 [notice] 1#1: start worker processes
2025/11/07 12:21:13 [notice] 1#1: start worker process 29
2025/11/07 12:21:13 [notice] 1#1: start worker process 30
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
```

kubectl logs uoc-54d7b5b8f-wgf8x --follow

```
ubuntu@ip-172-31-47-108:~$ kubectl logs uoc-54d7b5b8f-wgf8x --follow
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2025/11/07 12:21:12 [notice] 1#1: using the "epoll" event method
2025/11/07 12:21:12 [notice] 1#1: nginx/1.29.3
2025/11/07 12:21:12 [notice] 1#1: built by gcc 14.2.0 (Debian 14.2.0-19)
2025/11/07 12:21:12 [notice] 1#1: OS: Linux 6.14.0-1016-aws
2025/11/07 12:21:12 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2025/11/07 12:21:12 [notice] 1#1: start worker processes
2025/11/07 12:21:12 [notice] 1#1: start worker process 29
2025/11/07 12:21:12 [notice] 1#1: start worker process 30
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
10.244.0.1 - - [07/Nov/2025:12:26:35 +0000] "GET / HTTP/1.1" 200 615 "-" "curl/8.5.0" "-"
```

Esto confirma que **el balanceo de carga está funcionando correctamente**, ya que las solicitudes se distribuyen entre las tres réplicas del contenedor **altctrlsupr/app:dev**, garantizando una **alta disponibilidad y tolerancia a fallos**.

Bibliografía

- Jenkins. (s. f.). *Installing Jenkins*. Recuperado el 16 de noviembre de 2025, de https://www.jenkins.io/doc/book/installing/
- kubectl. (s. f.). *Instalar y configurar kubectl en Linux*. En Kubernetes Documentation. Recuperado el 16 de noviembre de 2025 de https://kubernetes.io/es/docs/tasks/tools/included/install-kubectl-linux/
- Minikube. (s. f.). *Get Started – Windows / x86-64 stable (.exe download)*. Recuperado el 16 de noviembre de 2025, de https://minikube.sigs.k8s.io/docs/start/?arch=%2Fwindows%2Fx86-64%2Fstable%2F.exe+download

Anexo

Anexo 1 - k8s.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: uoc
spec:
  replicas: 3
  selector:
    matchLabels:
      app: uoc
  template:
    metadata:
      labels:
        app: uoc
    spec:
      containers:
        - name: app
          image: altctrlsupr/app:dev
          imagePullPolicy: Never   # <- importante para usar la
imagen local de Minikube
          ports:
            - containerPort: 8080  # <-- cambia si tu app usa otro
puerto
          readinessProbe:
            httpGet: { path: "/", port: 8080 }
            initialDelaySeconds: 5
          livenessProbe:
            httpGet: { path: "/", port: 8080 }
            initialDelaySeconds: 10
---
apiVersion: v1
kind: Service
metadata:
  name: uoc
spec:
  type: LoadBalancer
  selector:
    app: uoc
  ports:
    - port: 80
      targetPort: 8080
```