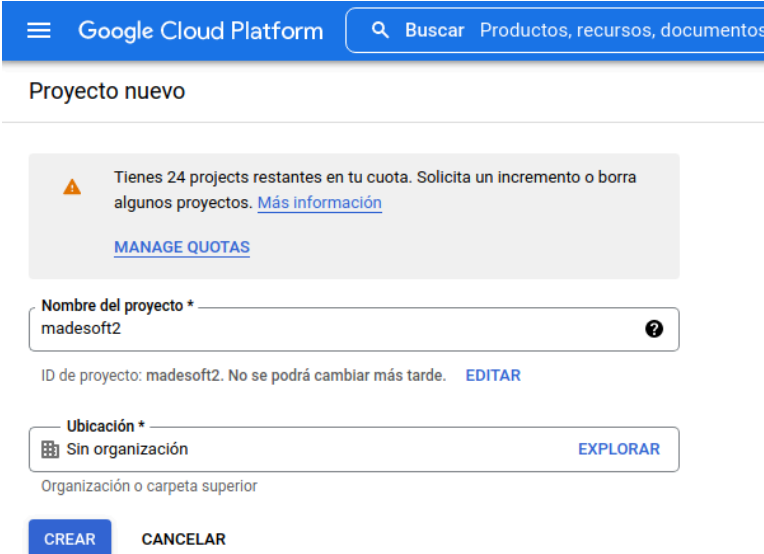


Paso a paso para la Implementación de despliegue continuo en google cloud proyecto inv-adc

Jaime Andrés Hurtado Giraldo
jaime.hurtado@correounivalle.edu.co
Escuela de Ingeniería de Sistemas y Computación
Universidad del Valle
Cali, Colombia

El siguiente procedimiento ha sido seleccionado luego de analizar diversas tecnologías para implementar en la nube de Google cloud el despliegue continuo para una aplicación basada en microservicios. Se ha depurado para hacer más práctica y replicable su implementación, de igual manera se ha realizado varias veces el procedimiento para probar su funcionamiento y vigente a la fecha junio 2 de 2023:

1. Ir a la página <https://console.cloud.google.com/> y loguearse con sus credenciales.
2. Ir al selector de proyectos y crear un nuevo proyecto



The screenshot shows the 'Proyecto nuevo' (New Project) page in the Google Cloud Platform console. At the top, there is a blue header with the Google Cloud Platform logo and a search bar. Below the header, the title 'Proyecto nuevo' is displayed. A warning message states: 'Tienes 24 projects restantes en tu cuota. Solicita un incremento o borra algunos proyectos. [Más información](#)'. Below this, there is a link 'MANAGE QUOTAS'. The main form has two sections: 'Nombre del proyecto *' with the text 'madesoft2' and a help icon, and 'Ubicación *' with a dropdown menu showing 'Sin organización' and an 'EXPLORAR' button. Below the form, there are two buttons: 'CREAR' (Create) and 'CANCELAR' (Cancel). At the bottom, it says 'ID de proyecto: madesoft2. No se podrá cambiar más tarde. [EDITAR](#)'.

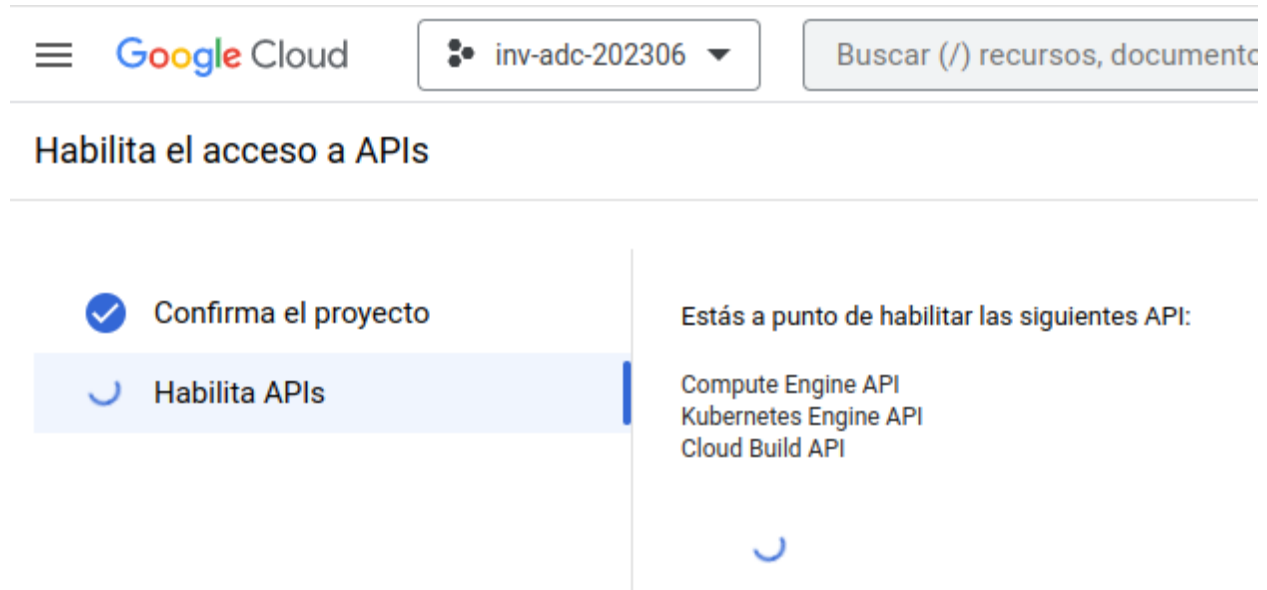
3. Ir al selector de proyectos y escoger el proyecto recién creado



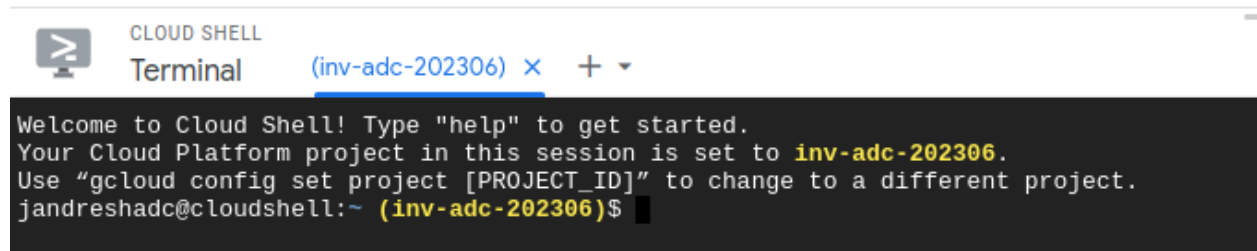
The screenshot shows the Google Cloud Platform project selector. It features the Google Cloud logo on the left and a dropdown menu on the right. The dropdown menu is open, showing the selected project 'inv-adc-202306' with a downward arrow next to it.

4. ir a Menu>Billing>Account management y seleccionar una cuenta de facturación para el proyecto. GCloud ofrece una prueba de 90 días con 300 USD de créditos.

5. Active mediante el siguiente enlace [Google Cloud Platform](#) y seleccionando el proyecto respectivo las siguientes API de GCloud para el proyecto:
 - a. Compute Engine
 - b. Kubernetes Engine
 - c. Cloud Build



6. Con el icono indicado active la terminal de GCloud e ingrese al shell del proyecto.



7. Seleccione la zona de cómputo a usar puede ayudarse con el mapa de la zona mas cercana a su territorio [Mapa](#), o con la documentación oficial de google. en nuestro caso seleccionamos la zona us-east1-d e ingresela al terminal de GCloud:
Comando:

```
gcloud config set compute/zone us-east1-d
```

Salida:

Updated property [compute/zone].

8. Cree una variable de entorno para el proyecto de GCloud:

Comando:

```
export GOOGLE_CLOUD_PROJECT=$(gcloud config get-value project)
```

Salida:

```
Your active configuration is: [...]
```

9. Clone el repositorio del proyecto:

Comando:

```
git clone https://github.com/jandresh/inv-adc/
```

Salida:

```
Receiving objects: 100% (154/154), 13.02 MiB | 30.29 MiB/s, done.
```

```
Resolving deltas: 100% (51/51), done.
```

10. Cree una cuenta de servicio

Comando:

```
gcloud iam service-accounts create jenkins-sa \
  --display-name "jenkins-sa"
```

Salida:

```
Created service account [jenkins-sa].
```

11. Adicione los permisos requeridos por la cuenta de servicio para manejar el cluster kubernetes y acceder a Gcloud compute engine.

Comando:

```
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
"serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccoun
t.com" \
  --role "roles/viewer"
```

```
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
"serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccoun
t.com" \
  --role "roles/source.reader"
```

```
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
"serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccoun
t.com" \
```

```
--role "roles/storage.admin"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
  "serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/storage.objectAdmin"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
  "serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/cloudbuild.builds.editor"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
  "serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/container.developer"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
  "serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/compute.instanceAdmin.v1"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
  "serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/compute.networkAdmin"

gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
  "serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/compute.securityAdmin"
```

```
gcloud projects add-iam-policy-binding $GOOGLE_CLOUD_PROJECT \
  --member
"serviceAccount:jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com" \
  --role "roles/iam.serviceAccountActor"
```

Salida:

```
- members:
  - serviceAccount:jenkins-sa@compdistuv.iam.gserviceaccount.com
    role: roles/storage.objectAdmin
- members:
  - serviceAccount:jenkins-sa@compdistuv.iam.gserviceaccount.com
    role: roles/viewer
etag: BwXMBgOClSA=
version: 1
```

12. Genere un archivo de credenciales jenkins-sa-key.json.

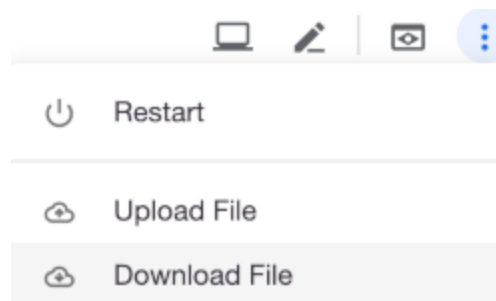
Comando:

```
gcloud iam service-accounts keys create ~/jenkins-sa-key.json \
  --iam-account
"jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com"
```

Salida:

```
created key [65669ec8eb0dd9405dc56890e6d1c7b6fed5e19f] of type
[json] as [/home/jandreshuv/jenkins-sa-key.json] for
[jenkins-sa@compdistuv.iam.gserviceaccount.com]
```

13. Descargue el archivo jenkins-sa-key.json generado en el paso anterior.



14. Crear una clave SSH para Cloud Shell. Ejecute los comandos separadamente.

Comandos:

```
ls ~/.ssh/ || mkdir ~/.ssh/
ls ~/.ssh/id_rsa.pub || ssh-keygen -N "" -f ~/.ssh/id_rsa
gcloud compute project-info describe \
    --format=json | jq -r '.commonInstanceMetadata.items[] |
select(.key == "ssh-keys") | .value' > sshKeys.pub
echo "$USER:$(cat ~/.ssh/id_rsa.pub)" >> sshKeys.pub
gcloud compute project-info add-metadata --metadata-from-file
ssh-keys=sshKeys.pub
```

Salida:

```
Updated Updated
[https://www.googleapis.com/compute/v1/projects/inv-adc-202306].
```

15. Instale el software packer para crear una imagen de una máquina virtual desde la cual se realizan compilaciones de docker y despliegues de prueba.

Comando:

```
wget
https://releases.hashicorp.com/packer/1.7.10/packer_1.7.10_linux_amd64.zip
unzip packer_1.7.10_linux_amd64.zip
```

Salida:

```
Archive:  packer_1.7.10_linux_amd64.zip
  inflating: packer
```

16. Cree el archivo Json de configuración de la imagen ubuntu. Si sale un error de imagen no encontrada listar las imagenes de linux (gcloud compute images list --filter ubuntu-os-cloud) y actualizar el comando

Comando:

```
export PROJECT=$(gcloud info --format='value(config.project)')
cat > jenkins-agent.json <<EOF
{
  "builders": [
    {
      "type": "googlecompute",
      "project_id": "$PROJECT",
      "source_image_family": "ubuntu-2004-lts",
      "source_image_project_id": "ubuntu-os-cloud",
      "zone": "us-east1-d",
      "disk_size": "10",
      "image_name": "jenkins-agent-{{timestamp}}",
```

```

        "image_family": "jenkins-agent",
        "ssh_username": "ubuntu"
    }
],
"provisioners": [
    {
        "type": "shell",
        "inline": ["sudo apt-get update && sudo apt-get upgrade -y",
"sudo add-apt-repository ppa:openjdk-r/ppa -y","sudo apt-get
update","sudo apt-get install ca-certificates-java openjdk-16-jre
-y", "java -fullversion", "curl -fsSL https://get.docker.com -o
get-docker.sh", "sudo sh get-docker.sh", "sudo usermod -aG docker
${USER}", "sudo apt-get install docker-compose -y"]
    }
]
}
EOF

```

Salida:

```

compose"]
> }
> ]
> }
> EOF

```

17. Ejecuta la compilación de packer para crear una imagen de ubuntu 2004. Revise en la información de salida que no haya errores en la compilación. Registre el nombre del agente que se imprime en la salida del comando. En este caso
jenkins-agent-1631704912

Comando:

```
./packer build jenkins-agent.json
```

Salida:

```

==> googlecompute: Deleting instance...
      googlecompute: Instance has been deleted!
==> googlecompute: Creating image...
==> googlecompute: Deleting disk...
      googlecompute: Disk has been deleted!
Build 'googlecompute' finished after 4 minutes 10 seconds.

==> Wait completed after 4 minutes 10 seconds

==> Builds finished. The artifacts of successful builds are:
--> googlecompute: A disk image was created: jenkins-agent-????????

```

18. Cree un clúster kubernetes de dos nodos con recursos tipo n1-standard-2.

Comando:

```

gcloud container clusters create jenkins-cd --num-nodes 2
--machine-type n1-standard-2 --cluster-version 1.23
--service-account
"jenkins-sa@$GOOGLE_CLOUD_PROJECT.iam.gserviceaccount.com"

```

Salida:

```

NAME: jenkins-cd
LOCATION: us-east1-d
MASTER_VERSION: 1.20.10-gke.301
MASTER_IP: 34.74.167.239
MACHINE_TYPE: n1-standard-2
NODE_VERSION: 1.20.10-gke.301
NUM_NODES: 2
STATUS: RUNNING

```

19. Adquiera las credenciales para el terminal la configuración y logueo del cliente kubectl para conexión con el cluster kubernetes recién creado.

Comando:

```

gcloud container clusters get-credentials jenkins-cd

```

Salida:

```

Fetching cluster endpoint and auth data.
kubeconfig entry generated for jenkins-cd.

```

20. Añada un administrador del cluster que le de autorización a Jenkins.

Comando:

```
kubectl create clusterrolebinding cluster-admin-binding
--clusterrole=cluster-admin --user=$(gcloud config get-value
account)
```

Salida:

```
min --user=$(gcloud config get-value account)
Your active configuration is: [cloudshell-15775]
clusterrolebinding.rbac.authorization.k8s.io/cluster-admin-binding
created
```

21. Ingrese al directorio del repositorio inv-adc.**Comando:**

```
cd inv-adc/
```

22. Instale Helm que es un manejador de paquetes de software del cluster, y Jenkins que es el servidor de integración y despliegue continuos. En el repositorio la carpeta Jenkins tiene el archivo de configuración con los plugins requeridos para el proyecto. En caso de instalar otro servidor de Jenkins, garantice que instala los plugins necesarios.**Comando:**

```
helm repo add jenkinsci https://charts.jenkins.io
helm repo update
helm install cd-jenkins -f jenkins/values.yaml jenkinsci/jenkins
--wait
```

Salida:

```
For more information on running Jenkins on Kubernetes, visit:
https://cloud.google.com/solutions/jenkins-on-container-engine
```

```
For more information about Jenkins Configuration as Code, visit:
https://jenkins.io/projects/jcasc/
```

NOTE: Consider using a custom image with pre-installed plugins

23. Configure el servidor Jenkins para que pueda desplegar en el cluster.**Comando:**

```
kubectl create clusterrolebinding jenkins-deploy
--clusterrole=cluster-admin --serviceaccount=default:cd-jenkins
```

Salida:

```
clusterrolebinding.rbac.authorization.k8s.io/jenkins-deploy created
```

24. Habilite el puerto 8080 del cluster para acceder al servidor Jenkins.

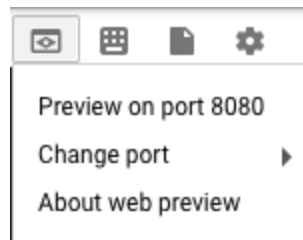
Comando:

```
export POD_NAME=$(kubectl get pods --namespace default -l
"app.kubernetes.io/component=jenkins-master" -l
"app.kubernetes.io/instance=cd-jenkins" -o
jsonpath="{.items[0].metadata.name}")
kubectl port-forward $POD_NAME 8080:8080 >> /dev/null 2>&1 &
```


Salida:

```
[1] 1157
```

25. Conéctese desde la parte superior derecha de la consola de GCloud al puerto 8080 para acceder a Jenkins



26. Adicione las credenciales del archivo json descargado, para ello ingrese por Dashboard>Manage Jenkins>Security>Credentials>(global)>Add Credentials. En el formulario el campo desplegable Kind seleccione la opción “Google Service Account from private key”, introduzca el project ID en el campo “Project Name”. El project Id se saca de la pantalla de inicio de GCloud. Seleccione el archivo JSON que tiene las credenciales de la cuenta de servicio. Oprima el botón “Create”



Información del proyecto

Nombre del proyecto
inv-adc-202306

Número del proyecto
345270621193

ID de proyecto
inv-adc-202306

[AGREGA PERSONAS A ESTE PROYECTO](#)

→ [Ir a la configuración del proyecto](#)

New credentials

Kind

Google Service Account from private key

Project Name ?

inv-adc-202306

☒ JSON key

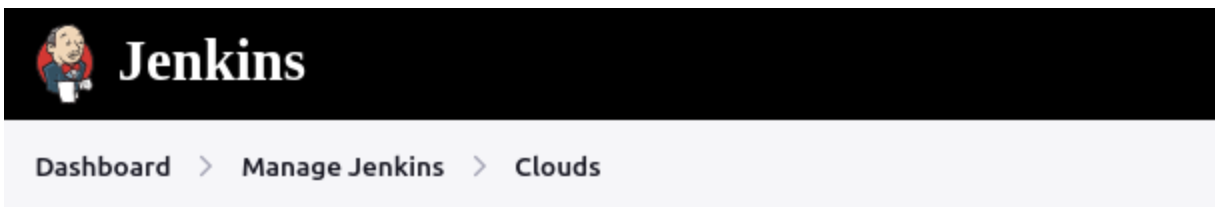
JSON key File ?


[Choose File](#) jenkins-sa-key.json

☐ P12 key

[Create](#)

27. Ir a “Manage Jenkins>System Configuration>Nodes and Clouds>Clouds>Add a new cloud> Google Compute Engine”



 Nodes

Clouds

[Add a new cloud ^](#)

Google Compute Engine

Kubernetes

[Save](#) [Apply](#)

28. Llenar el formulario de configuración de la nube Google Compute Engine teniendo en cuenta la orientación que se despliega al hacer click en los íconos de cada uno de los campos. A continuación se muestran un ejemplo de configuración:

Google Compute Engine

Name

gce

Project ID

tr-wdc-202306

Instance Cap

8

Service Account Credentials

tr-wdc-202306

ADD +

No delay provisioning

Instance Configurations

List of instance configurations that can be launched as Jenkins agents

General

Name Prefix

ubuntu-2004

Description

Ubuntu Agent

Node Retention Time

30

Usage

Use this node as much as possible

Labels

ubuntu-2004

Number of Executors

1

Launch Configuration

Launch Timeout

500

Use Internal IP

Ignore Jenkins Proxy?

Run as user

jenkins

Use Custom SSH Private Key?

Remote Location

Java Exec Path

java

Windows?

One-Shot

Enabled

Create snapshot?

Location

Region

us-east1

Zone

us-east1-d

Machine Configuration

Template to use

Advanced +

Edited

Machine Type

n1-standard-1

Preemptible?

Minimum Cpu Platform

Startup script

CPU

Networking

General

Available networks

Network

default

Subnetwork

default

Network tags

Attach External IP

Boot Disk

Image project

tr-wdc-202306

Image name

jenkins-agent-1685718677

Disk Type

pd-balanced

Size

20

Delete on termination?

IAM

Service Account E-mail

Save

Apply

29. En la parte inferior adicionar una nueva nube, Kubernetes, llenar el formulario de configuración teniendo en cuenta la orientación que se despliega al hacer click en los íconos de cada uno de los campos. A continuación se muestra un ejemplo de configuración:

The image displays four screenshots of the Jenkins configuration interface for a new Kubernetes cloud:

- Main Tab:** Shows the 'Name' field set to 'kubernetes'. The 'Kubernetes URL' is 'https://kubernetes.default'. The 'Kubernetes server certificate key' is empty. The 'Disable https certificate check' checkbox is checked. The 'Kubernetes Namespace' is 'default'. The 'Jenkins Docker Registry' is empty.
- Credentials Tab:** Shows the 'Credentials' dropdown set to 'in-adv-202306'. The 'Jenkins URL' is 'http://pod-jenkins:8080'. The 'Jenkins tunnel' is 'cd-jenkins-agent:50000'. The 'Connection Timeout' is '5'. The 'Read Timeout' is '15'. The 'Concurrency Limit' is '10'.
- Pod Labels Tab:** Shows the 'Pod Label' section with 'Key' set to 'jenkins' and 'Value' set to 'in-adv-app'. The 'Pod Retention' dropdown is set to 'Max connections to Kubernetes API' with a value of '12'. The 'Seconds to wait for pod to be running' is '600'.
- Advanced Tab:** Shows the 'Container Cleanup Timeout' set to '5'. The 'Transfer proxy related environment variables from controller to agent' checkbox is checked. The 'Restrict pipeline support to authorized folders' checkbox is checked. The 'Default Provider Template Name' is empty. The 'Pod Templates' section is empty.


30. Los despliegues en máquinas virtuales pueden requerir abrir los puertos en el Firewall de Google Cloud. Los despliegues en Kubernetes pueden desplegar automáticamente a IPs públicas junto con los respectivos puertos configurados. Para que los entornos de prueba se puedan acceder por la ip pública, configure los puertos 3000 de entrada y salida. En la consola de GCloud vaya a Menu>VPC network>Firewall>Create Firewall rule y llene el formulario por cada puerto de entrada o salida que requiera habilitar. Se muestra un ejemplo de habilitar puerto 5000 de entrada de otra aplicación.

31. Agregar una nueva tarea tipo multi branch, estas tareas están pendientes de cada rama del repositorio para implementar el pipeline. Acceda por Dashboard>New Item>Enter Item Name>Multibranch pipeline> OK

Enter an item name


pipeline

Required field




Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with an build system, and this can be even used for something other than software build.




Pipeline

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.




Multi-configuration project

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.



Folder


Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple thing of the same name as long as they are in different folders.



Multibranch Pipeline

Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:



Copy from

Type to autocomplete

OK

32. Finalmente configure solo el repositorio y el tiempo de escaneo de cambios en alguna rama, guarde y verifique la implementación

Dashboard > pipeline >

General **Branch Sources** Build Configuration Scan Multibranch Pipeline Triggers

Orphaned Item Strategy Health metrics Properties Pipeline Libraries Kubernetes

Git

Project Repository [?](#)

Credentials [?](#)

[Add](#)

Behaviors

Discover branches [?](#)

[Add](#)

Property strategy

[Add property](#)

[Add source](#)

Build Configuration

Mode

Script Path [?](#)

Scan Multibranch Pipeline Triggers

☒ Periodically if not otherwise run [?](#)

Interval [?](#)

Orphaned Item Strategy

[Save](#) [Apply](#) [Cancel](#)

Jobs (i.e. deleted branches) can be removed immediately or kept based on a desired strategy. If jobs will be removed as soon as Jenkins determines their associated SCM head

Build Queue (2)

part of pipeline » master #1

part of pipeline » development #1

Build Executor Status

1 Idle

2 Idle

Scan Multibranch Pipeline Log

Started

```
[Sun Feb 13 23:58:12 UTC 2022] Starting branch indexing...
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git ls-remote --symref -- https://github.com/jandresh/madesoft2 # timeout=10
Creating git repository in /var/jenkins_home/caches/git-e594b4146630928b2fb959142efc2f1e
> git init /var/jenkins_home/caches/git-e594b4146630928b2fb959142efc2f1e # timeout=10
Setting origin to https://github.com/jandresh/madesoft2
> git config remote.origin.url https://github.com/jandresh/madesoft2 # timeout=10
Fetching & pruning origin...
Listing remote references...
> git config --get remote.origin.url # timeout=10
> git --version # timeout=10
> git --version # 'git version 2.30.2'
> git ls-remote -h -- https://github.com/jandresh/madesoft2 # timeout=10
Fetching upstream changes from origin
> git config --get remote.origin.url # timeout=10
> git fetch --tags --force --progress --prune -- origin
+refs/heads/*:refs/remotes/origin/* # timeout=10
Checking branches...
Checking branch master
'Jenkinsfile' found
Met criteria
Scheduled build for branch: master
```

33. En la consola de google cloud revise los servicios para encontrar las ip públicas para acceder a la app. En el resultado se observa que los despliegues de entornos de desarrollo tienen un namespace “development” y los de producción un namespace “production”

Comando:

```
kubectl get services -A
```

Salida:

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE					
default	cd-jenkins	ClusterIP	10.3.248.62	<none>	8080/TCP
38m					
default	cd-jenkins-agent	ClusterIP	10.3.248.240	<none>	50000/TCP
38m					
default	kubernetes	ClusterIP	10.3.240.1	<none>	443/TCP
41m					
development	blog	LoadBalancer	10.3.248.134	34.138.119.117	
3000:30383/TCP	9m13s				
development	mongo	ClusterIP	10.3.245.32	<none>	27017/TCP
9m13s					
kube-system	default-http-backend	NodePort	10.3.246.35	<none>	80:30991/TCP
40m					
kube-system	kube-dns	ClusterIP	10.3.240.10	<none>	53/UDP,53/TCP
40m					
kube-system	metrics-server	ClusterIP	10.3.249.210	<none>	443/TCP
40m					

34. Cuando tenga actualizaciones que desee desplegar en producción haga un merge de la rama development a la rama canary. Esto debe hacerlo localmente desde su PC.

Comando:

```
git checkout canary
git merge development
git push origin canary
```

Salida:

```
Writing objects: 100% (2/2), 307 bytes | 307.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/jandresh/madesoft1.git
0233864..f024fee canary -> canary
```

Jenkins:

Pipeline canary

Full project name: pipeline/canary



Stage View

		Build&Test app	Container Publish	Test App form dockerHub	Deploy Developer	Deploy Canary	Deploy Production	Declarative: Post Actions
Average stage times:		1min 22s	46s	23s	0ms	36s	183ms	370ms
#2 Sep 21 18:10 2 commits		28s	48s	3s				
#1 Sep 21 17:43 No Changes								
		2min 16s	44s	43s		46s Failed	183ms Failed	370ms

Cluster Kubernetes comando "kubectl get services -A" se observa despliegue automatizado con el namespace production:

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
AGE					
cardiovasculares	corews	ClusterIP	10.3.251.176	<none>	5003/TCP
25h					
cardiovasculares	dbws	LoadBalancer	10.3.241.7	34.75.109.141	
5001:30162/TCP	25h				
cardiovasculares	metapubws	ClusterIP	10.3.244.166	<none>	5000/TCP
25h					
cardiovasculares	mongo	ClusterIP	10.3.253.106	<none>	
27017/TCP	25h				
cardiovasculares	mysql	ClusterIP	10.3.252.146	<none>	3306/TCP
25h					
cardiovasculares	orchestratorws	LoadBalancer	10.3.240.208	35.190.155.126	
5004:31261/TCP	25h				
cardiovasculares	preprocessingws	ClusterIP	10.3.246.75	<none>	5002/TCP
25h					
default	cd-jenkins	ClusterIP	10.3.247.74	<none>	8080/TCP
3d8h					
default	cd-jenkins-agent	ClusterIP	10.3.250.245	<none>	
50000/TCP	3d8h				
default	kubernetes	ClusterIP	10.3.240.1	<none>	443/TCP
3d9h					
development	corews	ClusterIP	10.3.244.84	<none>	5003/TCP
23h					
development	dbws	LoadBalancer	10.3.247.128	35.237.18.243	
5001:30965/TCP	23h				
development	metapubws	ClusterIP	10.3.245.50	<none>	5000/TCP
23h					
development	mongo	ClusterIP	10.3.250.161	<none>	
27017/TCP	23h				
development	mysql	ClusterIP	10.3.253.32	<none>	3306/TCP
23h					
development	orchestratorws	LoadBalancer	10.3.249.38	34.139.79.244	
5004:31001/TCP	23h				
development	preprocessingws	ClusterIP	10.3.255.212	<none>	5002/TCP
23h					
jandresh	corews	ClusterIP	10.3.241.220	<none>	5003/TCP
23h					
jandresh	dbws	LoadBalancer	10.3.245.101	34.148.47.228	
5001:32622/TCP	23h				
jandresh	metapubws	ClusterIP	10.3.255.193	<none>	5000/TCP
23h					
jandresh	mongo	ClusterIP	10.3.252.212	<none>	
27017/TCP	23h				
jandresh	mysql	ClusterIP	10.3.246.186	<none>	3306/TCP
23h					
jandresh	orchestratorws	LoadBalancer	10.3.253.217	35.190.180.15	
5004:32275/TCP	23h				
jandresh	preprocessingws	ClusterIP	10.3.242.248	<none>	5002/TCP
23h					
kube-system	default-http-backend	NodePort	10.3.255.213	<none>	
80:31427/TCP	3d9h				

kube-system	kube-dns	ClusterIP	10.3.240.10	<none>	
53/UDP,53/TCP	3d9h				
kube-system	metrics-server	ClusterIP	10.3.242.225	<none>	443/TCP
3d9h					
production	corews	ClusterIP	10.3.254.34	<none>	5003/TCP
2d19h					
production	dbws	LoadBalancer	10.3.254.198	34.148.109.5	
5001:31288/TCP	2d19h				
production	metapubws	ClusterIP	10.3.245.99	<none>	5000/TCP
2d19h					
production	mongo	ClusterIP	10.3.249.172	<none>	
27017/TCP	2d19h				
production	mysql	ClusterIP	10.3.252.0	<none>	3306/TCP
2d19h					
production	orchestratorws	LoadBalancer	10.3.252.49	35.237.37.139	
5004:30356/TCP	2d19h				
production	preprocessingws	ClusterIP	10.3.244.24	<none>	5002/TCP
2d19h					

35. Cuando haya probado en producción el despliegue canary actualice la rama master entonces haga un merge de la rama canary a la rama master. Esto debe hacerlo localmente desde su PC.

Comando:

```
git checkout master
git merge canary
git push origin master
```

Salida:

```
Writing objects: 100% (1/1), 251 bytes | 251.00 KiB/s, done.
Total 1 (delta 0), reused 0 (delta 0)
To https://github.com/jandresh/madesoft1.git
dd891c1..f88fa3f master -> master
```

Jenkins:

Pipeline master

Full project name: pipeline/master

 Recent Changes

Stage View

	Build&Test app	Container Publish	Test App form dockerHub	Deploy Developer	Deploy Canary	Deploy Production	Declarative: Post Actions
Average stage times:	1min 34s	53s	15s	0ms	0ms	36s	476ms
 Sep 21 18:22 8 commits	25s	43s	14s				
 Sep 21 17:43 No Changes	2min 42s	1min 3s	15s			49s  Failed	476ms