

Deploying en Jenkins

Continuous deployment and Continuous integration

A) CREAR Y CONECTARSE A UNA MÁQUINA EC2

Region

- us-east-1

1. Crea una máquina de EC2 en la cuenta de AWS

Sistema Operativo : Ubuntu Server 20.04

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 1: Choose an Amazon Machine Image (AMI) Cancel and Exit

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Q ubuntu

Search by Systems Manager parameter

Quick Start (8) |< < 1 to 8 of 8 AMIs > >|

My AMIs (0)	4 Ubuntu Server 20.04 LTS (HVM), SSD Volume Type - ami-042e8287309f5df03 (64-bit x86) / ami-0b75998a97c952252 (64-bit Arm)	Select
AWS Marketplace (631)	Free tier eligible 4 Ubuntu Server 20.04 LTS (HVM), EBS General Purpose (SSD) Volume Type . Support available from Canonical (http://www.ubuntu.com/cloud/services). Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
Community AMIs (36717)	4 Ubuntu Server 18.04 LTS (HVM), SSD Volume Type - ami-013f1736f8b11efb (64-bit x86) / ami-02ed82f3a38303e6f (64-bit Arm)	Select
<input type="checkbox"/> Free tier only ⓘ	Free tier eligible 4 Ubuntu Server 18.04 LTS (HVM), EBS General Purpose (SSD) Volume Type . Support available from Canonical (http://www.ubuntu.com/cloud/services). Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	<input checked="" type="radio"/> 64-bit (x86) <input type="radio"/> 64-bit (Arm)
	4 Deep Learning AMI (Ubuntu 18.04) Version 41.0 - ami-05f6982c11ca3027d MXNet-1.8.0 & 1.7.0, TensorFlow-2.4.1, 2.1.3 & 1.15.5, PyTorch-1.4.0 & 1.8.0, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA support. For fully managed experience, check: https://aws.amazon.com/sagemaker Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	Select 64-bit (x86)
	4 Deep Learning AMI (Ubuntu 16.04) Version 41.0 - ami-0b644137e2e3b1c23 MXNet-1.8.0 & 1.7.0, TensorFlow-2.4.1, 2.1.3 & 1.15.5, PyTorch-1.4.0 & 1.8.0, EI, Neuron, & others. NVIDIA CUDA, cuDNN, NCCL, Intel MKL-DNN, Docker, NVIDIA-Docker & EFA. For fully managed experience, check: https://aws.amazon.com/sagemaker Root device type: ebs Virtualization type: hvm ENA Enabled: Yes	Select 64-bit (x86)

Del tipo t2 small

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families Current generation Show/Hide Columns

Currently selected: t2.small (- ECUs, 1 vCPUs, 2.5 GHz, -, 2 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-
<input type="checkbox"/>	t2	t2.micro Free tier eligible	1	1	EBS only	-
<input checked="" type="checkbox"/>	t2	t2.small	1	2	EBS only	-
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-

Con las configuraciones de red por defecto

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

Number of instances	1	Launch into Auto Scaling Group
Purchasing option	<input type="checkbox"/> Request Spot instances	
Network	vpc-e119e59c (default) Create new VPC	
Subnet	No preference (default subnet in any Availability Zone) Create new subnet	
Auto-assign Public IP	Use subnet setting (Enable)	
Placement group	<input type="checkbox"/> Add instance to placement group	
Capacity Reservation	Open	
Domain join directory	No directory Create new directory	
IAM role	None Create new IAM role	
CPU options	<input type="checkbox"/> Specify CPU options	
Shutdown behavior	Stop	
Stop - Hibernate behavior	<input type="checkbox"/> Enable hibernation as an additional stop behavior	
Enable termination protection	<input type="checkbox"/> Protect against accidental termination	
Monitoring	<input type="checkbox"/> Enable CloudWatch detailed monitoring Additional charges apply.	
Tenancy	Shared - Run a shared hardware instance Additional charges will apply for dedicated tenancy.	
Elastic Inference	<input type="checkbox"/> Add an Elastic Inference accelerator Additional charges apply.	

Con un disco de 8 GB

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encryption ⓘ
Root	/dev/sda1	snap-0c8d535cd8fde4c4a	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. [Learn more](#) about free usage tier eligibility and usage restrictions.

Con el tag

Name : Jenkins

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 5: Add Tags

A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver.

A copy of a tag can be applied to volumes, instances or both.

Tags will be applied to all instances and volumes. [Learn more](#) about tagging your Amazon EC2 resources.

Key (128 characters maximum)	Value (256 characters maximum)	Instances ⓘ
<input type="text" value="Name"/>	<input type="text" value="Jenkins"/>	<input checked="" type="checkbox"/>
<p>Add another tag (Up to 50 tags maximum)</p>		

Con un nuevo grupo de seguridad que habilite el acceso por el puerto 22 desde

cualquier red y otra regla que habilite el acceso al puerto 8080 desde cualquier lugar.

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow internet traffic to reach your instance, add rule HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group
☐ Select an existing security group

Security group name:

Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin
Custom TCP	TCP	8080	Custom CIDR, IP or Security Group	e.g. SSH for Admin

Add Rule

Warning
 Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cuando creamos la máquina asegurémonos de crear una llave de ssh con el nombre de jenkins

Details. You can go back to edit changes for each section. Click **Launch** to assign a key pair to your instance and complete the launch process.

your instance's security. Your security group, launch-wizard-1, is open to the world. accessible from any IP address. We recommend that you update your security group rules to allow access from known IP addresses only. al ports in your security group to facilitate access to the application or service you're running, e.g., HTTP (80) for web servers. [Edit security groups](#)

uration is not eligible for the free usage tier. 's eligible for the free usage tier, check your AMI selection,

04 LTS (HVM), SSD Volume Type - ami-042e8287309f... TS (HVM),EBS General Purpose (SSD) Volume Type. Support a virtualization type: hvm

Launch wizard-1
 launch-wizard-1 created 2021-03-19T15:47:02.488-03:00

Protocol	Port Range	Source	Description
TCP	22	0.0.0.0/0	
TCP	8080	0.0.0.0/0	
TCP	8080	0.0.0.0/0	

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. [Learn more about removing existing key pairs from a public AMI.](#)

Create a new key pair

Key pair name

Download Key Pair

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

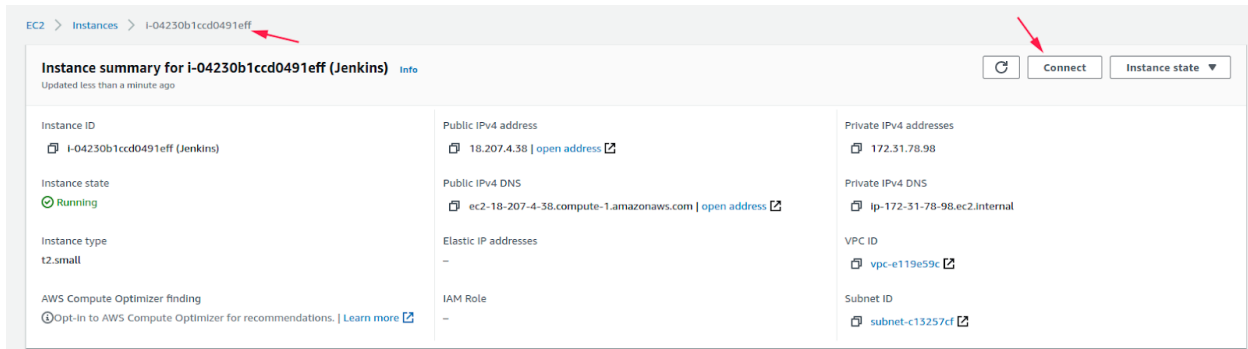
Cancel Launch Instances

Cancel Previous **Launch**

Descargar la llave de ssh en la máquina local

2. Conectarse a la máquina

Ingresa a la configuración de la máquina, luego en Connect.



Ubuntu

En el directorio donde se encuentra el archivo jenkins.pem ejecutar

```
chmod 400 jenkins.pem
```

Luego ejecutar en la consola

```
ssh -i "jenkins.pem" ubuntu@ec2-18-207-4-38.compute-1.amazonaws.com
```

EC2 > Instances > i-04230b1ccd0491eff > Connect to Instance

Connect to instance [Info](#)


Connect to your instance i-04230b1ccd0491eff (Jenkins) using any of these options




EC2 Instance Connect

Session Manager



SSH client


Instance ID

 i-04230b1ccd0491eff (Jenkins)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is jenkins1.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 `chmod 400 jenkins1.pem` 
4. Connect to your Instance using its Public DNS:
 `ec2-18-207-4-38.compute-1.amazonaws.com`

Example:

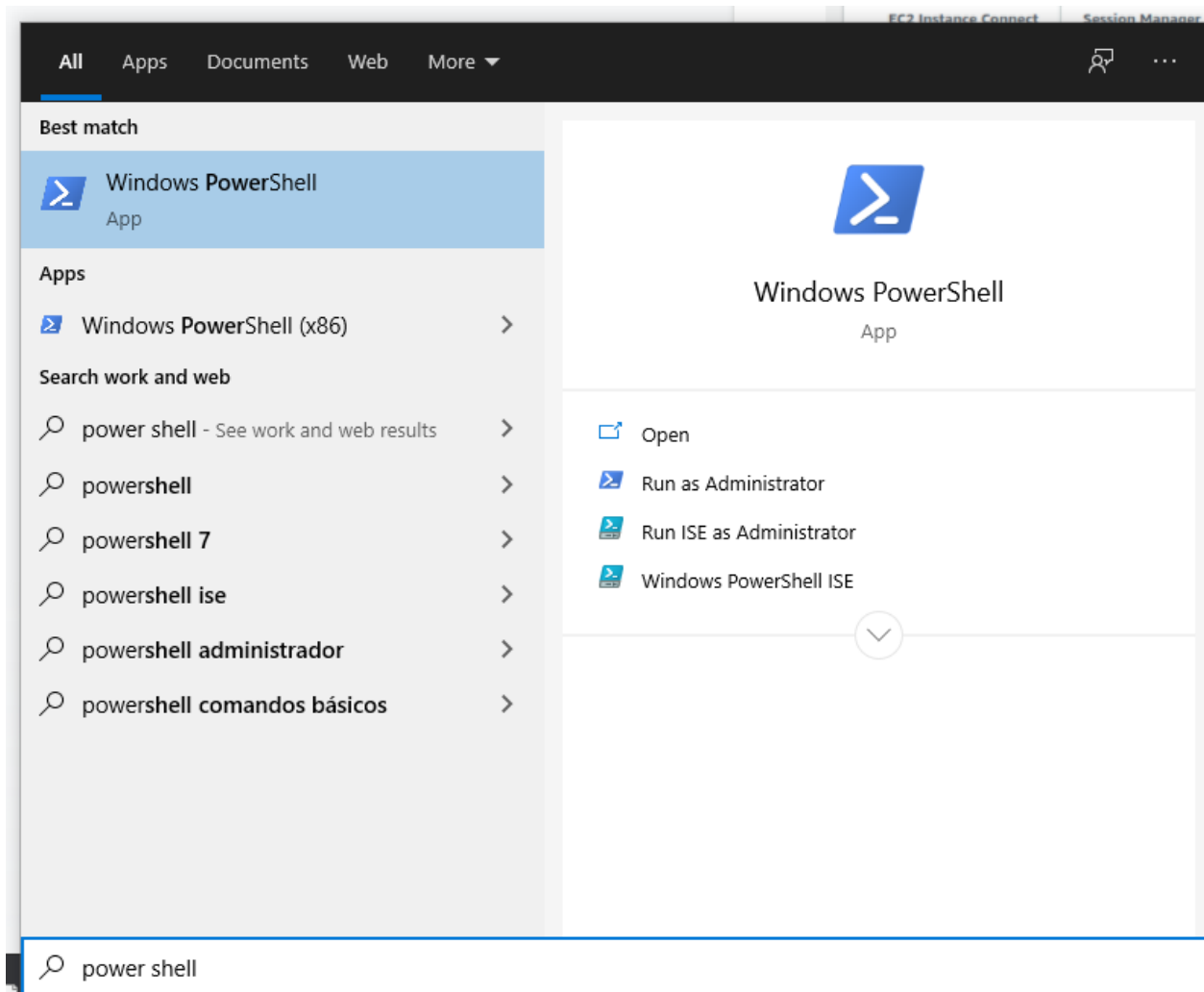
 `ssh -i "jenkins1.pem" ubuntu@ec2-18-207-4-38.compute-1.amazonaws.com` 

 **Note:** In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Cancel

Windows

Abrir power shell



Luego ejecutar en la consola

```
ssh -i "jenkins.pem" ubuntu@ec2-18-207-4-38.compute-1.amazonaws.com
```

```
PS C:\Users\test\Downloads> ssh -i "jenkins3.pem" ubuntu@ec2-18-204-56-2.compute-1.amazonaws.com
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-1038-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

System information as of Fri Mar 19 20:06:14 UTC 2021

System load:  0.0               Processes:           104
Usage of /:   16.4% of 7.69GB   Users logged in:    0
Memory usage: 11%              IPv4 address for eth0: 172.31.79.81
Swap usage:   0%

1 update can be installed immediately.
0 of these updates are security updates.
To see these additional updates run: apt list --upgradable

The list of available updates is more than a week old.
To check for new updates run: sudo apt update

Last login: Fri Mar 19 19:59:22 2021 from 186.122.180.199
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-79-81:~$
```

B) PRERREQUISITOS

1. Instalar unzip

```
sudo apt install unzip
```

2. Instalar cliente de AWS


```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o  
"awscliv2.zip"
```

```
unzip awscliv2.zip
```

```
sudo ./aws/install
```

3. Verifica instalación

```
aws --version
```

4. Instalar Helm

```
curl https://baltocdn.com/helm/signing.asc | sudo apt-key add -  
  
sudo apt-get install apt-transport-https --yes  
  
echo "deb https://baltocdn.com/helm/stable/debian/ all main" | sudo tee  
/etc/apt/sources.list.d/helm-stable-debian.list  
  
sudo apt-get update  
  
sudo apt-get install helm
```

5. Verificar instalación de helm

```
helm version
```

6. Configurar cliente

ATENCIÓN, Por razones de seguridad solicitar las Keys a los administradores

Ejecutar

Colocar Access Key ID Secret Access Key generadas

```
aws configure
```

```
AWS Access Key ID [*****C24K]:  
AWS Secret Access Key [*****IEuX]:  
Default region name [us-east-1]: us-east-1  
Default output format [json]: json
```

7. Instalar jq

```
sudo apt-get install jq -y
```

8. Instalar pip

```
sudo apt install python3-pip -y
```

9. Instalar eksctl

```
curl --silent --location  
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname -s)_amd64.tar.gz" | tar xz -C /tmp  
  
sudo mv /tmp/eksctl /usr/local/bin  
  
eksctl version
```

10. Instalar kubectl

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

```
sudo install -o root -g root -m 0755 kubectl /usr/local/bin/kubectl
```

```
kubectl version --client
```

11. Creación del cluster de Kubernetes

```
eksctl create cluster \
```

```
--name eks-mundos-e \  
--region us-east-1 \  
--with-oidc \  
--ssh-access \  
--ssh-public-key deployment-jenkins \  
--managed
```

C) REPOSITORIO, ACCESO Y CÓDIGO EN CODECOMMIT

1. Crear Repositorio CodeCommit

Empezaremos creando un repositorio [CodeCommit](#) para almacenar nuestra aplicación de ejemplo. Este repositorio almacenará nuestro código de aplicación y [Jenkinsfile](#).

```
aws codecommit create-repository --repository-name eksworkshop-app
```

2. Crear y configurar usuario IAM

Crearemos un usuario de IAM con nuestras credenciales HTTPS Git para que AWS CodeCommit clone nuestro repositorio y envíe confirmaciones adicionales. Este usuario necesita una política de IAM para acceder a CodeCommit.

```
aws iam create-user \  
--user-name borjas-prodolliet
```

```
aws iam attach-user-policy \  
--user-name borjas-prodolliet \  
--policy-name CodeCommitReadOnly
```

```
--policy-arn arn:aws:iam::aws:policy/AWSCodeCommitPowerUser
```

```
aws iam create-service-specific-credential \  
  --user-name borjas-prodolliet --service-name codecommit.amazonaws.com \  
  | tee /tmp/gituser_output.json
```

```
GIT_USERNAME=$(cat /tmp/gituser_output.json | jq -r  
'ServiceSpecificCredential.ServiceUserName')
```

```
GIT_PASSWORD=$(cat /tmp/gituser_output.json | jq -r  
'ServiceSpecificCredential.ServicePassword')
```

```
CREDENTIAL_ID=$(cat /tmp/gituser_output.json | jq -r  
'ServiceSpecificCredential.ServiceSpecificCredentialId')
```

3. Clonar el repositorio

El repositorio requerirá algo de código inicial, por lo que clonaremos el repositorio y agregaremos una aplicación Go simple.

```
sudo pip3 install git-remote-codecommit
```

```
git clone codecommit::us-east-1://eksworkshop-app
```

```
cd eksworkshop-app
```

4. Crear aplicación Go simple

`server.go` contiene nuestra sencilla aplicación.

```
package main

import (
    "fmt"
    "net/http"
)

func helloWorld(w http.ResponseWriter, r *http.Request){
    fmt.Fprintf(w, "Hello World")
}

func main() {
    http.HandleFunc("/", helloWorld)
    http.ListenAndServe(":8080", nil)
}
```

`server_test.go` contiene nuestras pruebas unitarias.

```
package main

import (
    "net/http"
    "net/http/httptest"
    "testing"
)

func Test_helloWorld(t *testing.T) {
    req, err := http.NewRequest("GET", "http://domain.com/", nil)
    if err != nil {
        t.Fatal(err)
    }

    res := httptest.NewRecorder()
```

```

    helloWorld(res, req)

    exp := "Hello World"
    act := res.Body.String()
    if exp != act {
        t.Fatalf("Expected %s got %s", exp, act)
    }
}

```

5. Crear Jenkinsfile

El `Jenkinsfile` contendrá nuestra declaración de canalización, los contenedores adicionales en nuestros pods de agentes de compilación y qué contenedor se utilizará para cada paso de la canalización.

```

pipeline {
    agent {
        kubernetes {
            yaml """
apiVersion: v1
kind: Pod
spec:
  containers:
  - name: golang
    image: golang:1.13
    command:
    - cat
    tty: true
            """
        }
    }
    stages {
        stage('Run tests') {

```

```
    steps {
      container('golang') {
        sh 'go test'
      }
    }
  }
  stage('Build') {
    steps {
      container('golang') {
        sh 'go build -o eksworkshop-app'
        archiveArtifacts "eksworkshop-app"
      }
    }
  }
}
```

6. Agregar la app al repositorio

Agregaremos el código a nuestro código, confirmaremos el cambio y luego enviaremos el código a nuestro repositorio.

```
git add --all && git commit -m "Initial commit." && git push
```


D) CREACIÓN DE LA CUENTA DE SERVICIO DE JENKINS

Crearemos una cuenta de servicio para que Kubernetes la otorgue a los pods si necesitan realizar acciones de la API de CodeCommit (por ejemplo, GetCommit, ListBranches). Esto permitirá que Jenkins responda a nuevos repositorios, ramas y confirmaciones.

```
eksctl create iamserviceaccount \
  --name jenkins \
  --namespace default \
  --cluster eks-mundos-e \
  --attach-policy-arn arn:aws:iam::aws:policy/AWSCodeCommitPowerUser \
  --approve \
  --override-existing-serviceaccounts
```

E) DEPLOY JENKINS

Instalar Jenkins

1. Crear values.yaml

Comenzaremos creando el `values.yaml` para declarar la configuración de nuestra instalación de Jenkins.

```
controller:
  resources:
```

```
requests:
  cpu: "1024m"
  memory: "4Gi"
limits:
  cpu: "4096m"
  memory: "8Gi"
javaOpts: "-Xms4000m -Xmx4000m"
servicePort: 80
serviceType: LoadBalancer
agent:
  Enabled: false
rbac:
  create: true
serviceAccount:
  create: false
  name: "jenkins"
```

2. Crear servidor Jenkins

Ahora usaremos el `helm` cli para crear el servidor Jenkins como lo hemos declarado en el `values.yaml` archivo.

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

# Extraer el password de Jenkins
kubectl exec --namespace default -it svc/cicd-jenkins -c jenkins --
/bin/cat /run/secrets/chart-admin-password && echo
```

El resultado de este comando le proporcionará información adicional, como la contraseña `admin` y la forma de obtener el nombre de host del ELB que se suministró.

Démosle algo de tiempo para aprovisionar y, mientras lo hacemos, veamos si los pods arrancan.

```
kubectl get pods -w
```

Debería ver los pods en estado `init`, `pending` o `running`.

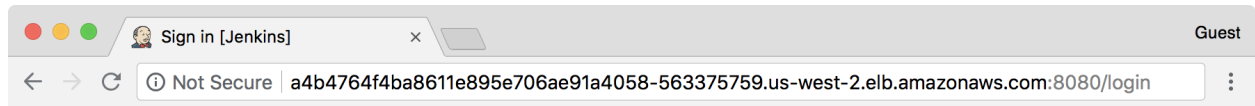
Una vez que esto cambie a `running` podemos obtener la dirección del `load balancer`.

```
export SERVICE_IP=$(kubectl get svc --namespace default cicd-jenkins
--template "{{ range (index .status.loadBalancer.ingress 0) }}{{ . }}{{ end
}}")

echo http://$SERVICE_IP/login
```

F) INICIANDO SESIÓN

Ahora que tenemos la dirección ELB de la instancia `jenkins`, podemos navegar a esa dirección en otra ventana.



Welcome to Jenkins!

Sign in

Desde aquí podemos iniciar sesión usando:

Nombre de usuario	Contraseña
administración	comando de abajo








```
printf $(kubectl get secret --namespace default cicd-jenkins -o  
jsonpath="{.data.jenkins-admin-password}" | base64 --decode);echo
```

El resultado de este comando le dará la contraseña predeterminada para su usuario `admin`. Inicie sesión en la pantalla `jenkins` de inicio de sesión con estas credenciales.

G) CONFIGURAR PROYECTOS MULTIBRANCH

Después de iniciar sesión en la consola web de Jenkins, estamos listos para agregar nuestro repositorio `eksworkshop-app`. Comience seleccionando `New Item` en el menú del lado izquierdo.




-  New Item
-  People
-  Build History
-  Manage Jenkins
-  My Views
-  Lockable Resources
-  New View

Establezca el nombre del elemento en `codecommit` y seleccione el tipo de elemento `AWS Code commit`.


Enter an item name

codecommit


» Required field

**Freestyle project**


This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any 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.

**AWS Code commit**

Scans a AWS Code commit for all repositories matching some defined markers.

**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 things 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.

OK

En su espacio de trabajo de Cloud9, ejecute los siguientes comandos para obtener su nombre de usuario y contraseña de Git.

```
echo $GIT_USERNAME
echo $GIT_PASSWORD
```

De vuelta a Jenkins. En la sección Proyectos, a la derecha de **Code Commit Credentials**, seleccione **Add** luego **CodeCommit**.

Child Scan Triggers Pipeline Libraries Kubernetes Automatic branch project triggering

Display Name

Description

[Plain text] [Preview](#)

Disable ☐ (No new builds within this Organization Folder will be executed until it is re-enabled)

Projects

AWS Code commit	URL	<input type="text" value="https://codecommit.us-east-1.amazonaws.com"/>
	Regex	<input type="text" value="*"/>
	AWS Credentials	<input type="text" value="- none -"/> Add
	Code Commit Credentials	<input type="text" value="- none -"/> Add
	Behaviours	<div><div>Discover</div><div><div>codecommit</div><div>Jenkins</div></div><div>Add</div></div>

Project Recognizers

Establezca **Username** y **Password** en los valores correspondientes del comando anterior y haga clic **Add**.

Add Credentials

Domain

Kind

Username

Password

ID

Description

[Add](#) [Cancel](#)


Confirme su región de AWS actual.


```
echo https://codecommit.$AWS_REGION.amazonaws.com
```

Copie ese valor en el campo **URL** del proyecto y seleccione su uso en el menú desplegable Credenciales de confirmación de código.

The screenshot shows the Jenkins configuration interface for an AWS Code commit project. The 'General' tab is active. The 'Projects' section lists the 'AWS Code commit' project. The 'URL' field is set to 'https://codecommit.us-east-2.amazonaws.com'. The 'Code Commit Credentials' dropdown is open, showing 'git-user-at-197520326489/*****' selected. The 'Behaviours' section has a 'Discover branches' button.

Seleccione **Save** en la parte inferior izquierda de la pantalla. Jenkins comenzará a ejecutar los pipelines en repositorios y branches que contengan un **Jenkinsfile**.

 **Jenkins**

Jenkins > codecommit > eksworkshop-app > master >

Up

Status

Changes

Build Now

View Configuration

Full Stage View

Pipeline Syntax

Build History

find

X

#1

Aug 28, 2020 9:53 PM

Atom feed for all

Atom feed for failures

Pipeline master

Full project name: codecommit/eksworkshop-app/master

Last Successful Artifacts

eksworkshop-app

7.11 MB view

Recent Changes

Stage View

Average stage times:
(Average full run time: ~1min 18s)

Declarative: Checkout SCM	Run tests	Build
14s	6s	3s

#1

Aug 28 14:53

No Changes

Permalinks

- [Last build \(#1\), 1 min 32 sec ago](#)
- [Last stable build \(#1\), 1 min 32 sec ago](#)
- [Last successful build \(#1\), 1 min 32 sec ago](#)
- [Last completed build \(#1\), 1 min 32 sec ago](#)

H) LIMPIAR

Para desinstalar Jenkins y limpiar la cuenta de servicio y el repositorio de CodeCommit, ejecute:

```
helm uninstall cicd

aws codecommit delete-repository \
```

```
--repository-name eksworkshop-app

aws iam detach-user-policy \
  --user-name git-user \
  --policy-arn arn:aws:iam::aws:policy/AWSCodeCommitPowerUser

aws iam delete-service-specific-credential \
  --user-name git-user \
  --service-specific-credential-id $CREDENTIAL_ID

aws iam delete-user \
  --user-name git-user

eksctl delete iamserviceaccount \
  --name jenkins \
  --namespace default \
  --cluster eksworkshop-eksctl

rm -rf ~/environment/eksworkshop-app
rm ~/environment/values.yaml

sudo pip uninstall -y git-remote-codecommit

eksctl delete cluster \
  --name eks-mundos-e
```