

Experto Universitario en DevOps & Cloud

Caso práctico 1.

Guía de apoyo

Índice

Visión global	3
Setup inicial cuenta AWS Academy	4
Acceso a consola de administración de AWS	9
Billing AWS	12
Región	13
Securización	14
Creación de instancia EC2 para Jenkins y Cloud9	16
Introducción de la práctica	41
Caso práctico 1. Apartado A	45
Resumen de la solución. Apartado A	46
Clonado repositorio de la práctica y copia en repositorio de alumno	48
Validación de interfaz de SAM CLI (Command Line Interface) y análisis de repositorio	52
Ejecución de proyecto en entorno local (SAM CLI)	54
Despliegue manual de aplicación SAM en Amazon Web Services	56
Creación de <i>pipelines</i> de Jenkins para despliegue de arquitectura completa	67
Caso práctico 1. Apartado B	81
Caso práctico 1. Apartado C	82
Anexo I. Entornos de ejecución	85
Anexo II . Referencias	91

Visión global

Desarrollo de un proyecto de integración (CI) y entrega continua (CD) de aplicaciones en Cloud. En este trabajo el alumno aplicará y desarrollará los conocimientos adquiridos a lo largo del curso.

El objetivo del presente trabajo es que el alumno defina el SCM (Software Configuration Management) y realice un *pipeline* de integración y entrega continua de aplicaciones en la nube partiendo de un cambio de software base. Se abordarán hitos de implementación de scripts de automatización de la operativa, reporte para monitorización del estado del correcto funcionamiento, pruebas que respondan al espectro o tipología requerida en el ejercicio de certificación de la calidad de un software, así como ficheros de configuración de sandboxes o entornos aislados de validación de pruebas determinadas.

En línea con los contenidos impartidos hasta el tiempo de la realización de caso práctico 1, servicios autogestionados de Amazon Web Services de uso recurrente, fundamentalmente dentro del entorno de Serverless¹, tales como **AWS Cloud9, API Gateway, EC2, AWS Lambda, AWS S3, EC2, DynamoDB e IAM** serán de uso recurrente en su realización.

¹ El concepto Serverless significa «sin servidor», aunque siempre existirá un servidor detrás, la idea es usar servicios gestionados que abstraiga al desarrollador de la operación de esos servidores y se haga foco en lo importante, en el desarrollo del producto y no de la operación.

Setup inicial cuenta AWS Academy

Configuración inicial de la cuenta de estudiante AWS Academy, requerida para disponer de la mejor experiencia de uso posible en el ejercicio de implementación del presente caso práctico. Aun contando con dilatada experiencia en el uso del ecosistema de servicios de AWS, se recomienda su realización. En primer lugar, se debe acceder al Laboratorio de AWS, creado en AWS Academy donde se ha dado de alta al alumno en un momento anterior al desarrollo de la práctica, accediendo a la URL:

<https://www.awsacademy.com/AcademyClasses>

Una vez en la plataforma, acceder al curso de AWS Academy Learner Lab – Associate donde se ha dado de alta al alumno.

The screenshot shows the AWS Academy dashboard with a sidebar on the left containing icons for Usuario, Cuadro de mando, Asignaturas, Calendario, Bandeja de entrada, Historial, and Ayuda. The main area is titled 'Cuadro de mando' and displays three course cards. The first card, 'AWS Academy Learner Lab - Associate ALLAv1-18018', is highlighted with a red box. It features the logo of the Universidad Nacional de La Rioja (uni) and the text 'Experto Universitario DevOps & Cloud'. The second card is 'AWS Academy Learner Lab - Foundation ALLFv1-10252' and the third is 'Educator Orientation [EN] EduOnEN'.

AWS Academy Learner Lab - Associate Services provides a long-running sandbox environment for ad hoc exploration of AWS services. Within this class, students will have access to a **restricted set of AWS services**. Not all AWS documentation walk-through or sample labs that operate in an AWS Production account will work in the sandbox environment. You will retain access to the AWS resources set up in this environment for the duration of this course. We limit your budget (\$100), so you should exercise caution to prevent charges that will deplete your budget too quickly. If you exceed your budget, you will lose access to your environment and lose all of your work.

Each session lasts for 4 hours by default, although you can extend a session to run longer by pressing the start button to reset your session timer. At the end of each session, any resources you created will persist. However, we automatically shut EC2 instances down. Other resources, such as RDS instances, keep running. Keep in mind that we do not stop some AWS features, so they can still incur charges between sessions. For example, an Elastic Load Balancer or a NAT. You may wish to delete those types of resources and recreate them as needed to test your work during a session. You will have access to this environment for the duration of the class they enrolled

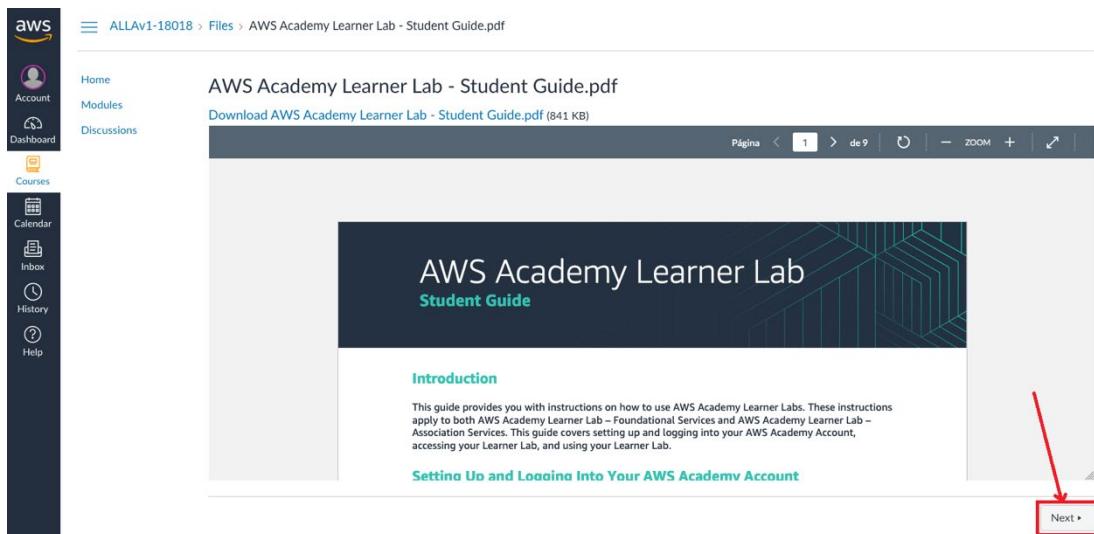
Una vez dentro del aula, se debe acceder al apartado de **Modules**, que es donde se encuentra el acceso al laboratorio.

AWS Academy Learner Lab - Associate Services provides a long-running sandbox environment for ad hoc exploration of AWS services. Within this class, students will have access to a **restricted set of AWS services**. Not all AWS documentation walk-through or sample labs that operate in an AWS Production account will work in the sandbox environment. You will retain access to the AWS resources set up in this environment for the duration of this course. We limit your budget (\$100), so you should exercise caution to prevent charges that will deplete your budget too quickly. If you exceed your budget, you will lose access to your environment and lose all of your work.

Each session lasts for 4 hours by default, although you can extend a session to run longer by pressing the start button to reset your session timer. At the end of each session, any resources you created will persist. However, we automatically shut EC2 instances down. Other resources, such as RDS instances, keep running. Keep in mind that we do not stop some AWS features, so they can still incur charges between sessions. For example, an Elastic Load Balancer or a NAT. You may wish to delete those types of resources and recreate them as needed to test your work during a session. You will have access to this environment for the duration of the class they enrolled

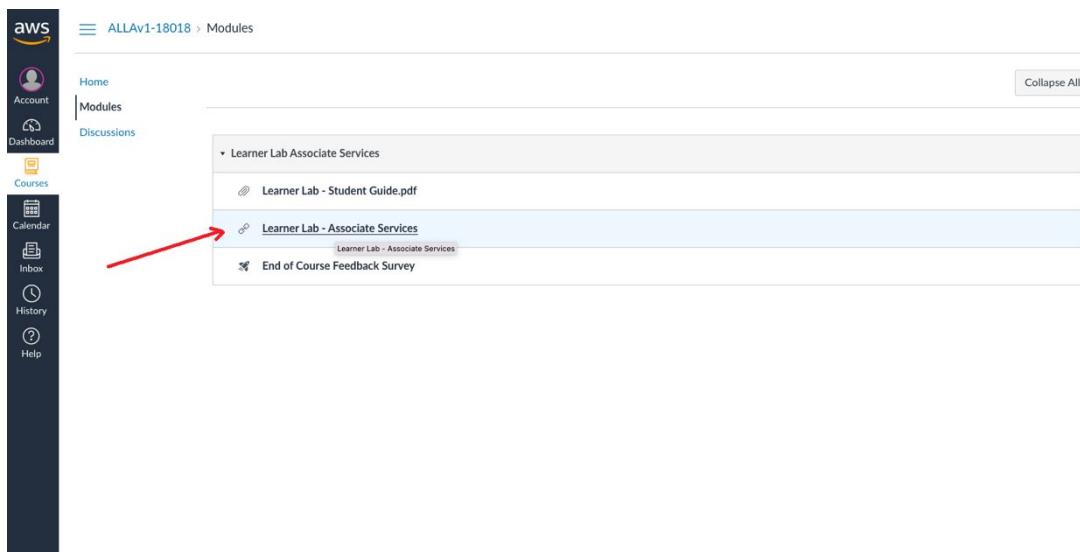
Dentro de **Modules**, se debe hacer clic en el enlace **Learner Lab – Associate Services**, para acceder al laboratorio.

IMPORTANTE: Previamente al acceso al modelo de la sesión de AWS (Learner Lab – Associates Services), será obligatorio acceder al módulo previo inicial y aceptar las condiciones de servicio tras su lectura detenida. El nombre de este módulo es ‘AWS Academy Learner Lab – Student Guide.pdf0, y las condiciones a aprobar figurarán tras al clic sobre el botón ‘Next’:



The screenshot shows a PDF document titled "AWS Academy Learner Lab - Student Guide.pdf". The document has a dark header with the title and a light footer containing sections like "Introduction" and "Setting Up and Logging Into Your AWS Academy Account". At the bottom right, there is a "Next" button with a red arrow pointing to it, indicating where to click to proceed.

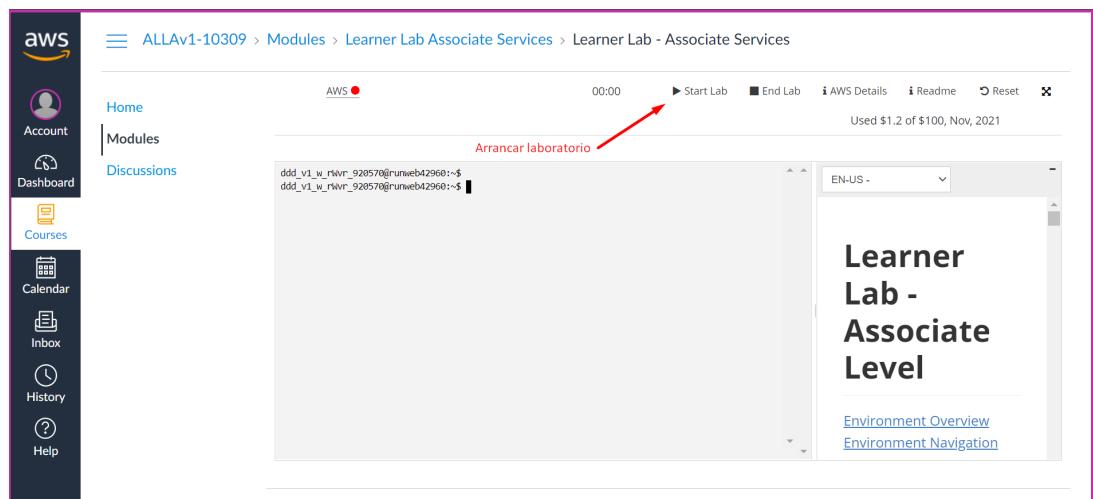
Una vez aprobadas las condiciones, ya podremos dirigirnos nuevamente al módulo que contiene el módulo de laboratorio:



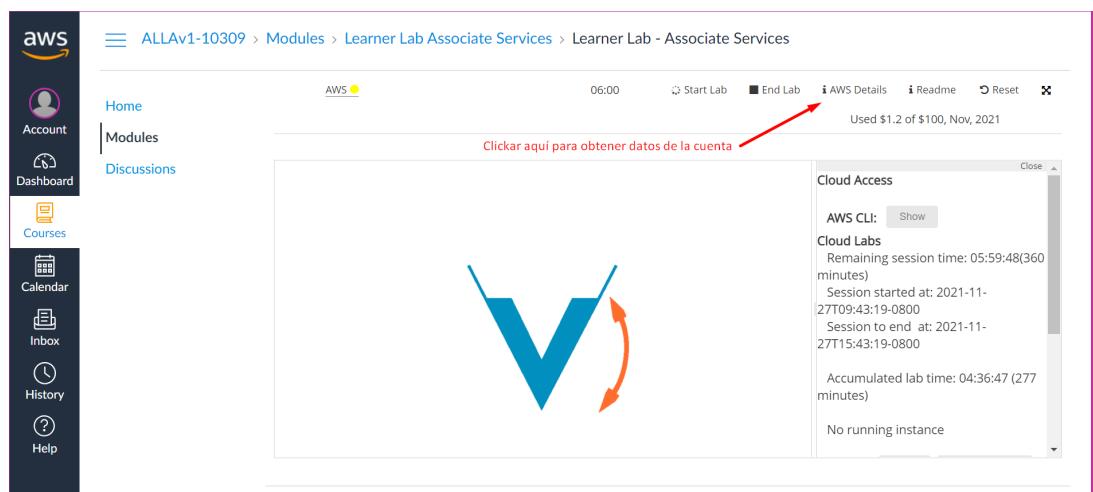
The screenshot shows the "Modules" section of the AWS Academy interface. It lists several modules, with one module titled "Learner Lab Associate Services" highlighted by a red arrow. This module contains sub-items like "Learner Lab - Student Guide.pdf" and "Learner Lab - Associate Services".

Una vez en el laboratorio, el alumno ya puede arrancar el laboratorio para poder trabajar. Es importante leerse la documentación asociada en inglés, para entender

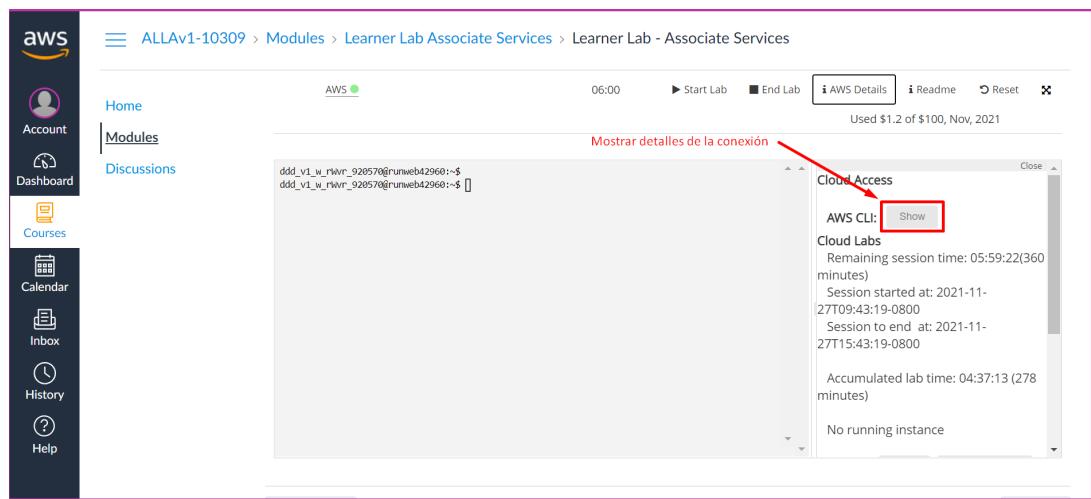
las restricciones que se aplican a la cuenta de AWS Academy. El alumno también deberá de tener en cuenta que los recursos creados en la cuenta de Academy no se destruyen mientras los créditos no se agoten, pero si se apagarán al cerrarse la sesión (entorno a seis horas aproximadamente), para ahorrar costes en el laboratorio. Una vez revisada la documentación de **Learner Lab – Associate Level**, se puede arrancar la cuenta haciendo clic en **Start Lab**:



En este momento, el logotipo del proveedor de la cuenta comenzará a girar, indicando así que está desplegándose la nueva cuenta en AWS. A partir de aquí ya se puede trabajar normalmente con la cuenta a través del navegador o bien a través de la consola que ofrece AWS Academy.

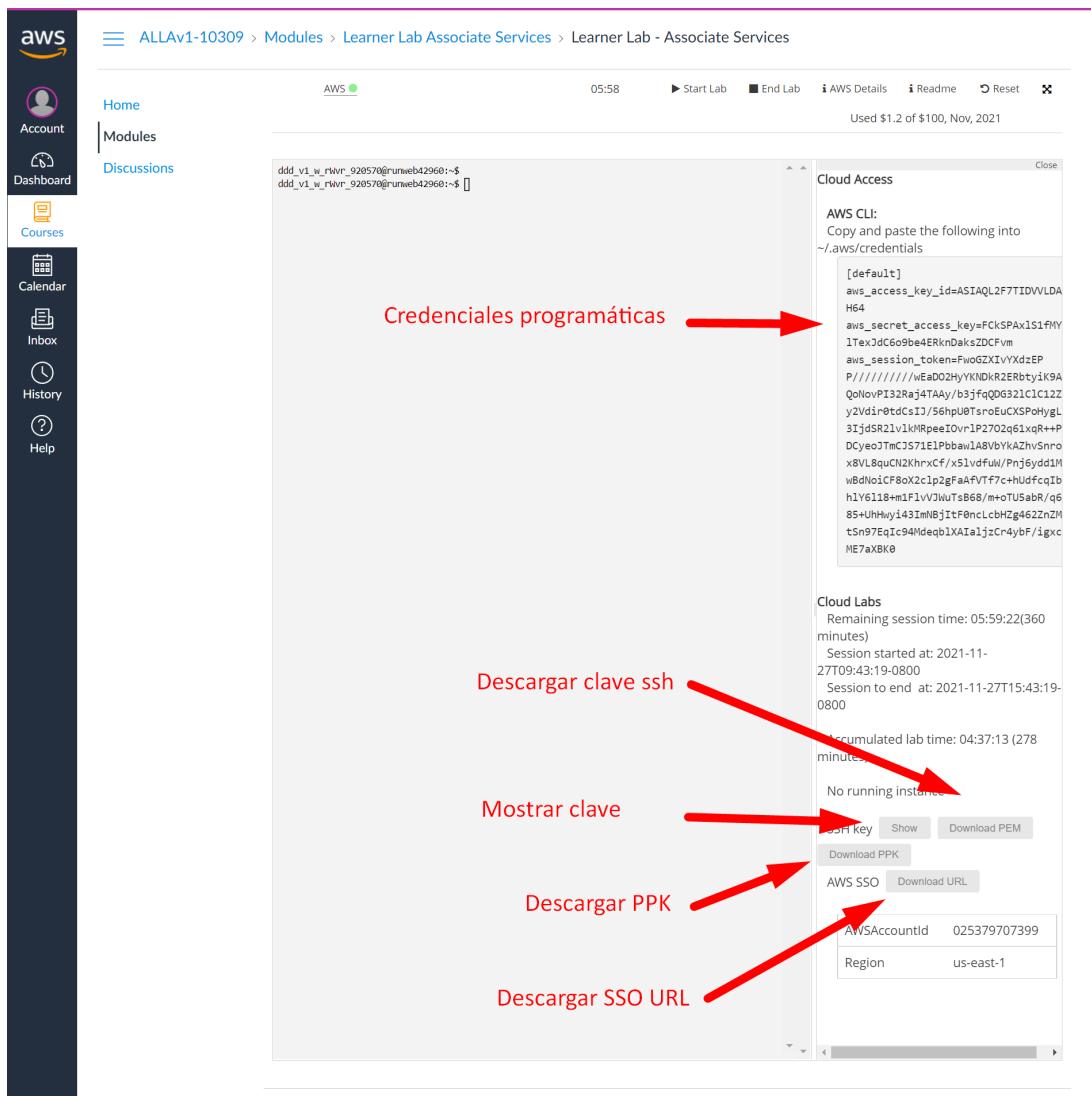


Los datos más relevantes que también refleja la interfaz del laboratorio son los datos de conexión a la cuenta de AWS, mostrados en el menú de la derecha. En este menú se ofrece la información necesaria tanto para conectarse a través de un enlace a la consola de AWS, como las credenciales programáticas de AWS (temporales). Para ver las credenciales, seleccionar el botón **Show**. Con ellas se podrá acceder a la cuenta de manera programática allí donde el alumno lo necesite.



Además de las credenciales programáticas, también se puede obtener la siguiente información, relevante para el desarrollo de la práctica, como son:

- ▶ **Claves SSH**, para poder acceder a las instancias EC2 que se levanten en la cuenta de AWS. Se pueden tanto descargar como mostrar. Para su uso en Linux.
- ▶ **Clave PPK**, para poder acceder a las instancias EC2 que se levanten en la cuenta de AWS. Se pueden tanto descargar como mostrar. Para su uso en Windows, con clientes como Putty.
- ▶ **Download url**, para poder descargar el enlace al portal de Single Sign-On (SSO) de la cuenta de AWS. Este fichero es de vital importancia, para poder trabajar desde la consola. Por tanto, el alumno **debe descargarse el fichero**, para poder trabajar como se describe a continuación.



Acceso a consola de administración de AWS

Una vez nos situemos dentro del módulo del laboratorio, deberemos de aguardar a que figure en verde el estado de creación de la sesión de AWS. Para acceder a la consola de administración, habrá que hacer clic sobre el enlace de 'AWS' (tal como figura en la siguiente captura de pantalla), momento en el que se nos abrirá una pestaña nueva para el acceso a la misma (**IMPORTANTE: Revisión en navegadores web de bloqueo de ventanas emergentes**):

The screenshot shows the AWS Learner Lab interface. At the top, there's a navigation bar with 'ALLAv1-1...' > 'Modules' > 'Learner La...' > 'Learner Lab - Associate Services'. Below the navigation is a sidebar with links: Home, Modules (which is selected), Discussions, Courses, Calendar, Inbox, History, and Help. The main area has a terminal window showing a command prompt: 'ddd_v1_w_p2W_1202454@runweb54532:~\$'. To the right of the terminal is a sidebar titled 'Learner Lab - Associate Level' containing a list of links related to environment overview, navigation, and various AWS services like EC2, S3, and Lambda.

Consola AWS

The screenshot shows the AWS console homepage. A blue banner at the top reads: 'La nueva página de inicio de la consola de AWS sustituirá pronto su experiencia actual. A partir de junio de 2022, la nueva página de inicio de la consola de AWS reemplazará su experiencia actual. Cambie ahora para personalizar la página de inicio de la consola y ver información valiosa. Más información o háganos saber lo que piensa.' Below the banner, the main title 'Consola de administración de AWS' is displayed, along with sections for 'Servicios de AWS', 'Cree una solución', and 'Lance una máquina virtual', 'Cree una aplicación web', and 'Diseñe con servidores virtuales'.

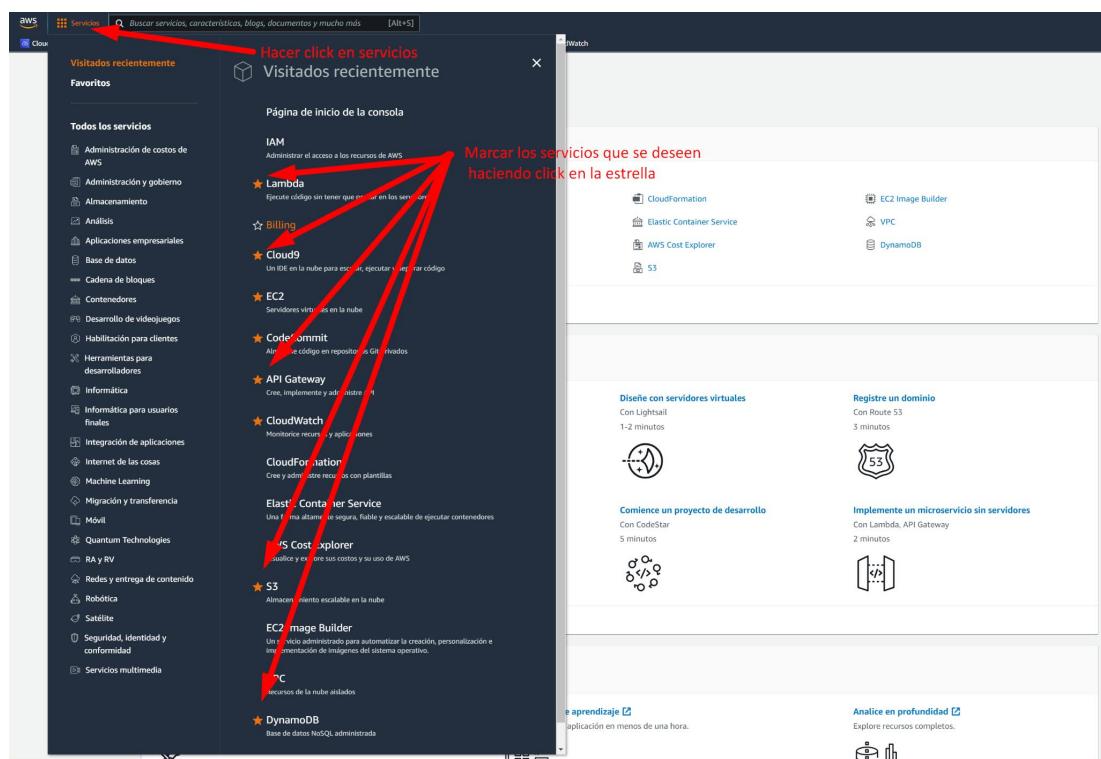
Consola de administración de AWS

The screenshot shows the AWS console homepage with the new start page layout. It features a sidebar on the left with 'Servicios de AWS' (Services) and 'Cree una solución' (Create a solution). The main area includes sections for 'Lance una máquina virtual' (Launch a virtual machine), 'Cree una aplicación web' (Create a web application), and 'Diseñe con servidores virtuales' (Design with virtual servers). To the right, there are two boxes: 'Nueva página de inicio de la consola de AWS' (New AWS console start page) and 'Manténgase conectado a sus recursos de AWS en cualquier lugar' (Stay connected to your AWS resources anywhere). The bottom of the page includes a footer with links for comments, search, and various AWS services.

Desde la consola de administración del proveedor Cloud **Amazon Web Services** (en adelante, [AWS](#)) se puede acceder al catálogo de servicios de múltiple espectro, orientados estos a abordar todas las etapas del ciclo de vida cualquiera solución profesional: despliegue de aplicaciones web, procesamiento distribuido y analítica avanzada, bases de datos de distinta naturaleza, cadenas de bloques, Internet de las Cosas, etc. Una vez introducimos las credenciales de acceso a la cuenta creada para el presente fin, puede observarse de un simple vistazo accesos directos para el uso de los servicios habituales, así como distinta documentación, material didáctico y

laboratorios guiados de aprendizaje con distintos niveles de profundización o dificultad.

Para el desarrollo de la siguiente práctica se aconseja hacer uso de la herramienta de PIN, situada en la barra superior, para poder tener un acceso directo a los servicios que van a ser utilizados durante el desarrollo de la práctica. Para ello, simplemente hay que hacer *click* en el botón de servicios y marcar la estrella del servicio deseado.

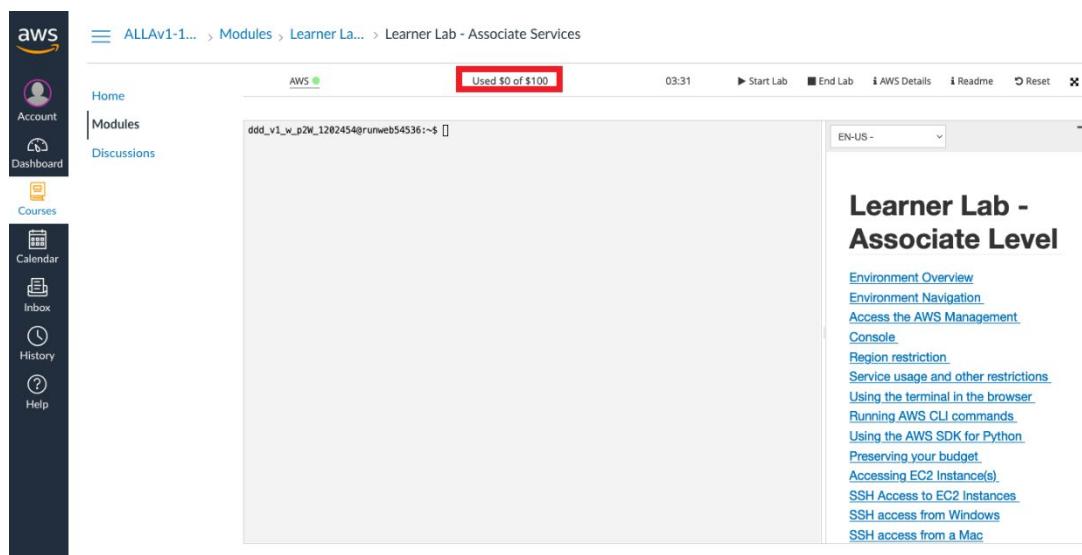


- ▶ [Cloud9](#)
- ▶ [CloudWatch](#)
- ▶ [API Gateway](#)
- ▶ [EC2](#)
- ▶ [Lambda](#)
- ▶ [S3](#)
- ▶ [DynamoDB](#)
- ▶ [IAM](#)

Para más detalle de cada uno de estos servicios, revisar los contenidos específicos del curso previo a la realización del caso práctico, así como los hipervínculos incluidos para cada servicio. Además, es importante revisar aquellos [servicios que están disponibles](#) dentro de la cuenta de AWS Academy. **AWS Academy no ofrece todos los servicios disponibles**, por tanto, alguna parte de la práctica requerirá de soluciones “imaginativas” que harán frente a las restricciones de la cuenta para poder desarrollar esta adecuadamente.

Billing AWS

Puesto que el servicio de facturación se encuentra deshabilitado en las cuentas de tipo Academy, el *billing* se ha de monitorizar a través del laboratorio de acceso de AWS Academy:



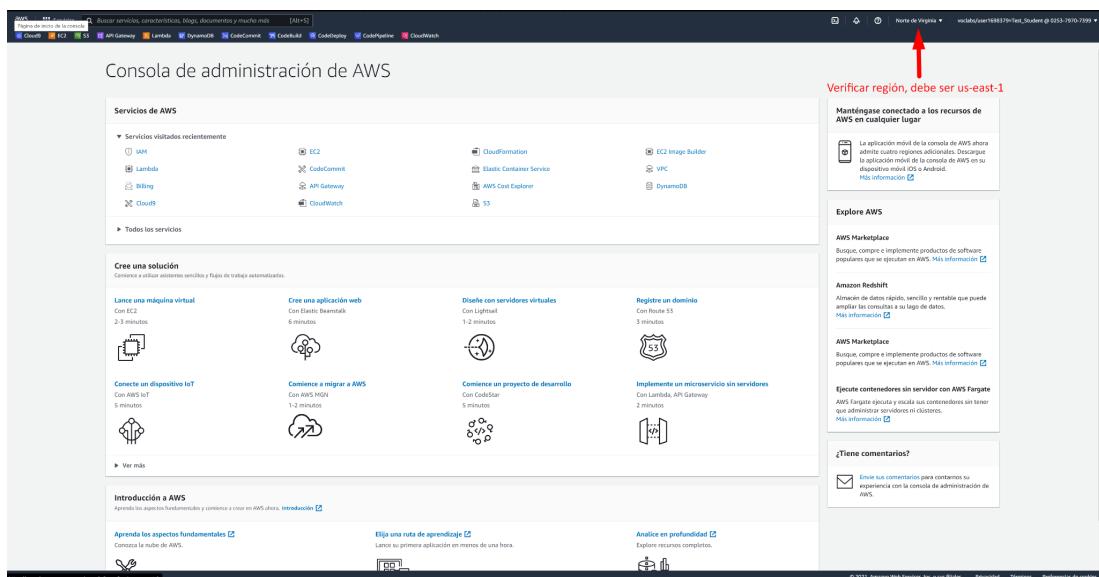
Importante **reseñar la necesidad de hacer una revisión frecuente del incremento de gasto durante el progreso de realización de la práctica**, fiel reflejo del uso de los servicios desde la vista anteriormente descrita. Recomendamos encarecidamente, **hacer caso del límite de uso establecido por alumno, haciendo un correcto uso de los servicios indicados**, no sobredimensionando los recursos computacionales o de

almacenamiento requeridos, así como otros que se alejen del propósito del caso práctico.

En caso de aproximarse al total de la cuantía asignada para el alumno, aspecto que deberá de comunicar al coordinador docente para de manera extraordinaria valorar una ampliación que posibilite su completa realización.

Región

Los servicios de AWS están hospedados en varias ubicaciones de todo el mundo. Dichas ubicaciones se componen de regiones, zonas de disponibilidad y Zonas locales. Cada región es un área geográfica independiente. Cada región tiene varias ubicaciones aisladas conocidas como zonas de disponibilidad. Las Zonas locales le proporcionan la capacidad de colocar recursos, como computación y almacenamiento, en varias ubicaciones más cercanas a sus usuarios finales. Los recursos no se replican en las regiones, a menos que usted decida hacerlo específicamente. Para el desarrollo de la práctica, se seleccionará la región del **Norte de Virginia (us-east-1)**, por cuestiones de limitación de la cuenta AWS Academy. **Por tanto, es imprescindible realizar el desarrollo de la práctica en esta región y sólo en ésta.**



Securización

En este setup inicial es importante configurar los roles y usuarios que van a ser necesarios para el desarrollo de la parte práctica. Para ello, habría de crearse un rol de instancia EC2, necesario en la creación y destrucción de recursos, sin necesidad de usar credenciales en la propia instancia. Parte de estos puntos, junto a otros muchos relativos a la securización (buenas prácticas en el no uso del usuario *root*, activación de la autenticación multifactor (MFA), entre otros) se encuentran disponibles dentro de la [guía de buenas prácticas](#) de AWS. En la pantalla de inicio del servicio de IAM, se puede observar el estado de la securización del servicio y de la cuenta.

Le damos la bienvenida a Identity and Access Management (IAM)

Hemos detectado los siguientes errores al procesar su solicitud:

- ✗ User: arn:aws:sts::411649862274:assumed-role/vocstartsoft/user818159@ruben.galeano@comunidadunir.net** is not authorized to perform: iam:ListAccountAliases on resource: * with an explicit deny

Recursos de IAM

Usuarios: 3	Roles: 38
Grupos: 0	Proveedores de identidad: 0
Políticas administradas por el cliente: 12	

Estado de seguridad

⚠ Activar MFA en la cuenta raíz	1 de 4 completado.
✓ Crear usuarios de IAM individuales	
⚠ Utilizar grupos para asignar permisos	
⚠ Aplicar una política de contraseñas de IAM	

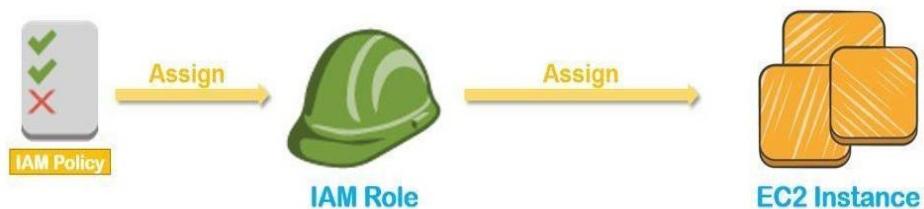
En este caso, por las restricciones impuestas por la plataforma Academy, no se podrán crear usuarios, ni roles, ni credenciales ni usar servicios como Security Token Service, por lo que no se podrá habilitar el MFA, ni aplicar todas las recomendaciones de buenas prácticas. Aun así, es importante leer la guía para comprenderlas y tener en consideración todos los posibles escenarios de peligro que sobre la cuenta puedan acontecer cuando se trabaja en AWS.

Usuarios

Como se ha indicado no se podrán crear usuarios en la cuenta de AWS, por lo que ninguna acción será requerida para el desarrollo de esta.

Roles

Para poder trabajar adecuadamente, se ha de crear un rol de mínimo permiso de EC2, de obligatorio cumplimiento para la creación de todo recurso que implícitamente requiere del despliegue de una instancia de máquina virtual durante el desarrollo de la práctica.



En el laboratorio de AWS Academy tampoco se podrán crear roles para adjuntar a las instancias EC2. Para conceder permisos sobre los servicios de AWS, el laboratorio AWS Academy provee de un rol llamado **LabRole** para todos los servicios que se van a usar en la práctica, y un perfil de instancia EC2 específico, de nombre

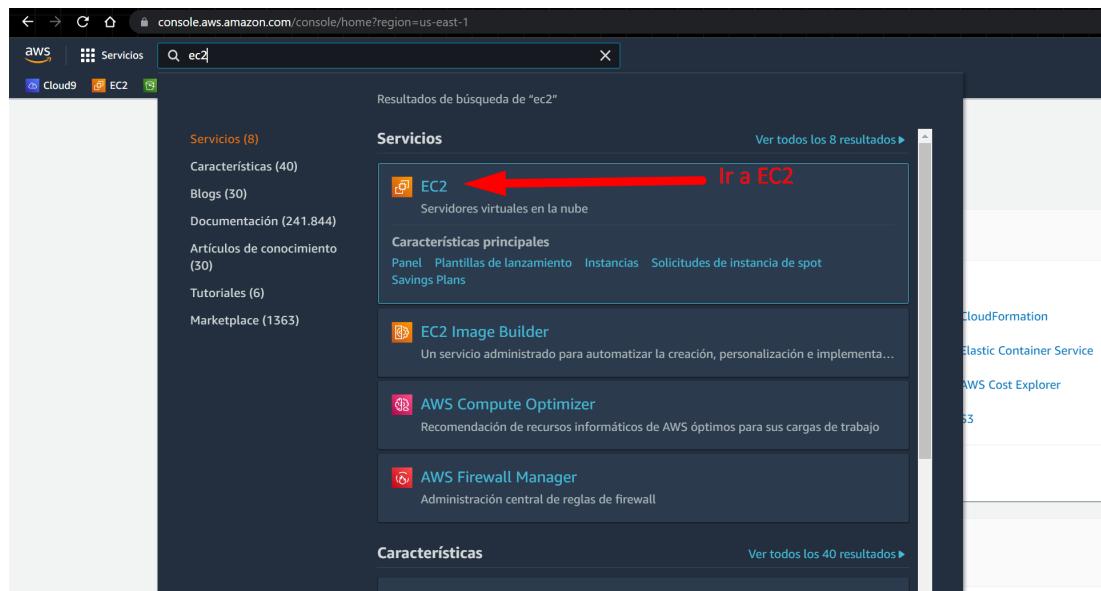
LabInstanceProfile. Adicionalmente, también se habrán creado una clave SSH para acceder a las instancias EC2, denominada **vockey**, que será la destinada a usarse en el desarrollo de la práctica.

Creación de instancia EC2 para Jenkins y Cloud9

Creación instancia EC2

Para lanzar una instancia de Ubuntu:

- ▶ En la consola de EC2, elegir **AMIs** (a la izquierda), cambiar el desplegable junto a la barra de búsqueda a **Public Images**, seleccionar la barra de búsqueda y elegir **Alias del Propietario**, luego elegir **amazon**.



Imágenes de Amazon Machine Image (AMI) (164590) Información

Imágenes públicas

<input type="checkbox"/>	Name	ID de AMI	Nombre de AMI	Origen
<input type="checkbox"/>	-	ami-d06ef2c7	zzxy	073544286138/zzxy
<input type="checkbox"/>	-	ami-0fe1c415618f32b9	zz0305	103066814212/zz0305
<input type="checkbox"/>	-	ami-024fc608af8f886bc	Zytblmg010422	925975727637/Zytblmg010422
<input type="checkbox"/>	-	ami-0fc5e52ba2aedb00d	zx-test-0.0.2	643962689773/zx-test-0.0.2
<input checked="" type="checkbox"/>	2. Seleccionar Imágenes Públicas	ami-a51376b2	Filtrar por las propietarias de amazon	Zurmo CRM on windows power...
<input type="checkbox"/>	-	ami-5c98fe23	zUMIs_25052018	7955868670709/zUMIs_25052018
<input type="checkbox"/>	-	ami-0f94959d22bd09289	zUMIs_2.4.0f-hvm	7955868670709/zUMIs_2.4.0f-hvm
<input type="checkbox"/>	-	ami-05c223683acf98e82	zUMIs_2.0-release-hvm	7955868670709/zUMIs_2.0-release-hvm
<input type="checkbox"/>	-	ami-7a987d04	zUMIs_2.0-release	7955868670709/zUMIs_2.0-release
<input type="checkbox"/>	-	ami-039069f90e87b3f18	zulu-jre-ami-2018q4-win16-c5c...	aws-marketplace/zulu-jre-ami-2018q4
<input type="checkbox"/>	-	ami-0515a52f71573c20a	zulu-jre-ami-2018q4-rhel76-41...	aws-marketplace/zulu-jre-ami-2018q4
<input type="checkbox"/>	-	ami-08d343eb21ac852b3	zulu-jdk-ami-2018q4-win16-c...	aws-marketplace/zulu-jdk-ami-2018q4
<input type="checkbox"/>	-	ami-0f650d429f6ebb24f	zulu-jdk-ami-2018q4-rhel76-e2...	aws-marketplace/zulu-jdk-ami-2018q4
<input type="checkbox"/>	-	ami-08fb0c5fa53515a70c	zulu-ami-2018q3-win16-ccce6...	aws-marketplace/zulu-ami-2018q3-wi...

1 2 3 4 5 6 7 ... 3292 < > ⌂

AMI New Catalogo de AMI Elastic Block Store

I1. Ir al menú de AMI
Seleccionar una AMI

- Volver a elegir Añadir filtro en la barra de búsqueda y elegir **Origen**, luego escribir **amazon/Cloud9Ubuntu**.

Imágenes de Amazon Machine Image (AMI) (15883) Información

Imágenes públicas

<input type="checkbox"/>	Name	ID de AMI	Nombre de AMI	Origen
<input type="checkbox"/>	-	ami-11ca2d78	-	aws-toolkit-for-eclipse-amis-us/tomca
<input type="checkbox"/>	-	ami-205fba49	-	ec2-public-images/fedora-core4-i386-
<input type="checkbox"/>	-	ami-20b65349	-	ec2-public-images/fedora-core4-base..
<input type="checkbox"/>	-	ami-215fba48	-	ec2-public-images/fedora-core4-base-
<input type="checkbox"/>	-	ami-225fba4b	-	ec2-public-images/fedora-core4-apach
<input type="checkbox"/>	-	ami-22b6534b	-	ec2-public-images/fedora-core4-mysq
<input type="checkbox"/>	-	ami-235fba4a	-	ec2-public-images/getting-started-v1.0
<input type="checkbox"/>	-	ami-23b6534a	-	ec2-public-images/fedora-core4-apach
<input type="checkbox"/>	-	ami-2547a34c	-	ec2-public-images/fedora-8-x86_64-b..
<input type="checkbox"/>	-	ami-255fba4c	-	ec2-public-images/fedora-core4-mysq
<input type="checkbox"/>	-	ami-25b6534c	-	ec2-public-images/fedora-core4-apach
<input type="checkbox"/>	-	ami-26b6534f	-	ec2-public-images/developer-image.m
<input type="checkbox"/>	-	ami-2a5fba43	-	ec2-public-images/fedora-8-x86_64-b..

1 2 3 4 5 6 7 ... 318 < > ⌂

AMI New Catalogo de AMI

Añadir filtro de Origen = **amazon/Cloud9Ubuntu**

- A continuación, ordenar el conjunto de resultados por el nombre de la AMI y elegir la más reciente (por ejemplo, Cloud9Ubuntu-YYYY-MMDDTHH-MM donde la marca de fecha es la más reciente):

Imágenes de Amazon Machine Image (AMI) (1296) Información

Nombre de AMI

Ordenar por Nombre de AMI y elegir la más reciente

Name	ID de AMI	Nombre de AMI	Origen
-	ami-099bc67a2500003c1	Cloud9Ubuntu-2022-04-28T13...	amazon/Cloud9Ubuntu-2022-04-28T1
-	ami-0a3ebdc9cdc8bde9e	Cloud9Ubuntu-2022-04-28T10...	amazon/Cloud9Ubuntu-2022-04-28T1
-	ami-03ac1356c97bbfcce	Cloud9Ubuntu-2022-04-26T13...	amazon/Cloud9Ubuntu-2022-04-26T1
-	ami-0d65597f50324b4c	Cloud9Ubuntu-2022-04-26T09...	amazon/Cloud9Ubuntu-2022-04-26T0
-	ami-064378f0c6bb6ce4d	Cloud9Ubuntu-2022-04-26T03...	amazon/Cloud9Ubuntu-2022-04-26T0
-	ami-0a0bc0528dfb10bf1	Cloud9Ubuntu-2022-04-25T16...	amazon/Cloud9Ubuntu-2022-04-25T1
-	ami-0b787c7cad7d9574	Cloud9Ubuntu-2022-04-25T15...	amazon/Cloud9Ubuntu-2022-04-25T1
-	ami-02aa5afb88fe440	Cloud9Ubuntu-2022-04-25T14...	amazon/Cloud9Ubuntu-2022-04-25T1
-	ami-0e4e813081cbf1a75	Cloud9Ubuntu-2022-04-25T13...	amazon/Cloud9Ubuntu-2022-04-25T1
-	ami-01655c92ba13d5342	Cloud9Ubuntu-2022-04-22T11...	amazon/Cloud9Ubuntu-2022-04-22T1
-	ami-0454e5ha8d026d203	Cloud9Ubuntu-2022-04-21T20...	amazon/Cloud9Ubuntu-2022-04-21T2
-	ami-0e54bd1b1fd37ef9	Cloud9Ubuntu-2022-04-21T14...	amazon/Cloud9Ubuntu-2022-04-21T1

Seleccionar una AMI

- Elegir la casilla de verificación junto a una y seleccionar **Lanzar instancia a partir de imagen.**

Imágenes de Amazon Machine Image (AMI) (1/1296) Información

Nombre de AMI

Lanzar instancia a partir de una AMI

Name	ID de AMI	Origen	Propietario	Alias del propietario	Visibilidad	Estado	Fecha de creación	Plataforma
<input checked="" type="checkbox"/> ami-099bc67a2500003c1	Cloud9Ubuntu-2022-04-28T13...	amazon/Cloud9Ubuntu-2022-04-28T1...	amazon	32794444948	Público	Disponible	Thu Apr 28 2022 14:00 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0a3ebdc9cdc8bde9e	Cloud9Ubuntu-2022-04-28T10...	amazon/Cloud9Ubuntu-2022-04-28T1...	amazon	32794444948	Público	Disponible	Thu Apr 28 2022 12:53:53 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-03ac1356c97bbfcce	Cloud9Ubuntu-2022-04-26T13...	amazon/Cloud9Ubuntu-2022-04-26T1...	amazon	32794444948	Público	Disponible	Tue Apr 26 2022 15:10:49 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0d65597f50324b4c	Cloud9Ubuntu-2022-04-26T09...	amazon/Cloud9Ubuntu-2022-04-26T0...	amazon	32794444948	Público	Disponible	Tue Apr 26 2022 17:48:03 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-064378f0c6bb6ce4d	Cloud9Ubuntu-2022-04-26T03...	amazon/Cloud9Ubuntu-2022-04-26T0...	amazon	32794444948	Público	Disponible	Tue Apr 26 2022 09:30:36 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0a0bc0528dfb10bf1	Cloud9Ubuntu-2022-04-25T16...	amazon/Cloud9Ubuntu-2022-04-25T1...	amazon	32794444948	Público	Disponible	Mon Apr 25 2022 12:12:47 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0b787c7cad7d9574	Cloud9Ubuntu-2022-04-25T15...	amazon/Cloud9Ubuntu-2022-04-25T1...	amazon	32794444948	Público	Disponible	Mon Apr 25 2022 16:09:44 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-02aa5afb88fe440	Cloud9Ubuntu-2022-04-25T14...	amazon/Cloud9Ubuntu-2022-04-25T1...	amazon	32794444948	Público	Disponible	Mon Apr 25 2022 17:13:18 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-01655c92ba13d5342	Cloud9Ubuntu-2022-04-25T13...	amazon/Cloud9Ubuntu-2022-04-25T1...	amazon	32794444948	Público	Disponible	Mon Apr 25 2022 16:09:59 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0454e5ha8d026d203	Cloud9Ubuntu-2022-04-21T20...	amazon/Cloud9Ubuntu-2022-04-21T2...	amazon	32794444948	Público	Disponible	Fri Apr 22 2022 13:42:00 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0d4e813081cbf1a75	Cloud9Ubuntu-2022-04-21T14...	amazon/Cloud9Ubuntu-2022-04-21T1...	amazon	32794444948	Público	Disponible	Thu Apr 21 2022 23:15:47 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0057e9316648effb	Cloud9Ubuntu-2022-04-21T10...	amazon/Cloud9Ubuntu-2022-04-21T1...	amazon	32794444948	Público	Disponible	Thu Apr 21 2022 13:54:28 GMT+0200 (...	Linux/UNIX
<input type="checkbox"/> ami-0f00ad57404ab106	Cloud9Ubuntu-2022-04-12T09...	amazon/Cloud9Ubuntu-2022-04-12T0...	amazon	32794444948	Público	Disponible	Tue Apr 12 2022 13:26:07 GMT+0200 (...	Linux/UNIX

ID de la AMI: ami-099bc67a2500003c1

Seleccionar instancia más moderna de Cloud9

Detalles | Almacenamiento | Etiquetas

Lanzar instancia a partir de una AMI

- Acto seguido, será recomendable hacer clic sobre el botón de regresar a la interfaz antigua de despliegue de instancias de EC2:

Se le ha agregado la nueva experiencia de lanzamiento. Obtenga más información sobre esta experiencia o envíenos sus comentarios. Puede volver a la versión anterior si no lo desea.

Volver a la experiencia antigua

Lanzar una instancia

Amazon EC2 le permite crear máquinas virtuales, o instancias, que se ejecutan en la nube de AWS. Comience rápidamente siguiendo los sencillos pasos que se indican a continuación.

Nombre y etiquetas

Nombre: p. ej., Mi servidor web

Agregar etiquetas adicionales

Imagenes de aplicaciones y sistemas operativos (Amazon Machine Image)

Una AMI es una plantilla que contiene la configuración de software (sistema operativo, servidor de aplicaciones y aplicaciones) necesaria para lanzar la instancia. Busque o examine las AMI si no ve lo que busca a continuación.

AMI del catálogo | Inicio rápido

Amazon Machine Image (AMI)

Cloud9Ubuntu-2022-04-28T13-38
ami-099bc67a250003c1

Buscar más AMI

Incluidas las AMI de AWS, Marketplace y la comunidad

Publicado	Arquitectura	Virtualización	Tipo de dispositivo	Habilitado para ENA
2022-04-28T14:14:00.00Z	x86_64	hvm	raíz ebs	Sí

Resumen

Número de instancias: 1

Imagen de software (AMI): Cloud9 Cloud9Ubuntu AMI ami-099bc67a250003c1

Tipo de servidor virtual (tipo de instancia): t2.micro

Firewall (grupo de seguridad): Nuevo grupo de seguridad

Almacenamiento (volúmenes): 1 volumen(es): 10 GiB

Nivel gratuito: El primer año incluye 750 horas de uso de instancias t2.micro (o t3.micro en las regiones en las que t2.micro no esté disponible) en las AMI del nivel gratuito al mes, 30 GiB de almacenamiento de EBS, 2 millones de E/S, 1 GB de instantáneas y 100 GB de ancho de banda a Internet

Cancelar Lanzar instancia

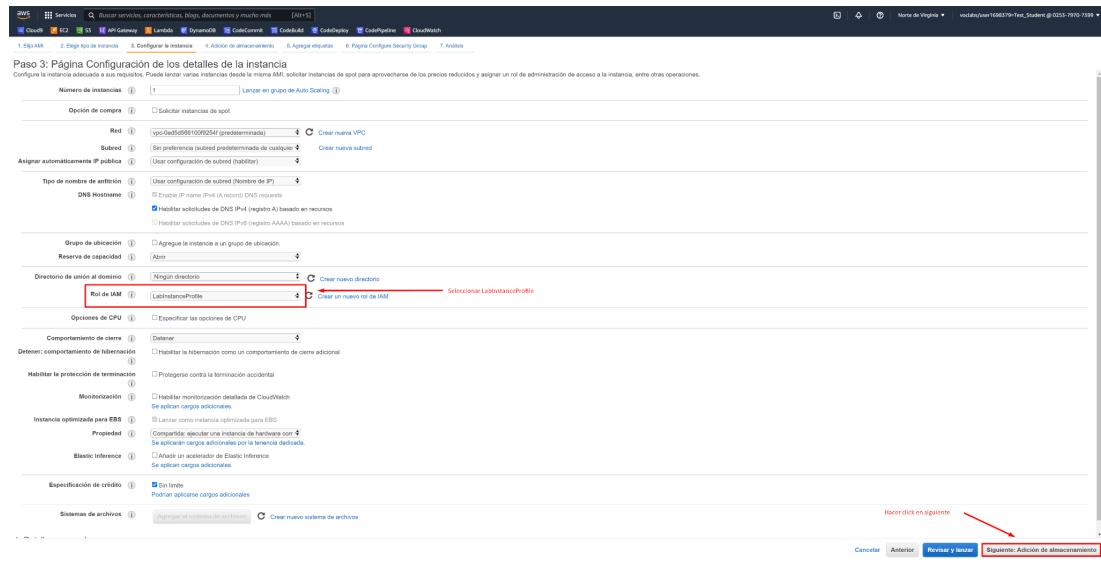
- Seleccionar el tipo de instancia y continuar con la configuración de la instancia haciendo clic en **Siguiente: Página configuración de los detalles de la instancia**. Inicialmente se recomienda usar **T3.micro**:

Paso 2: Página Choose an Instance Type

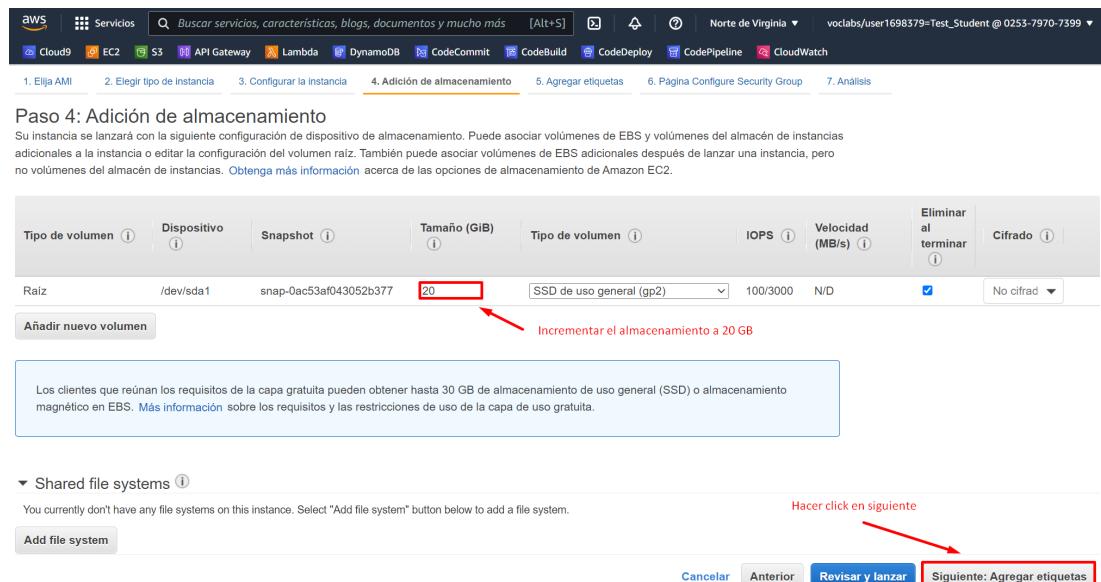
t2	t2.large	2	8	EBS solo	-	De bajo a moderado	Si
t2	t2.xlarge	4	16	EBS solo	-	Moderada	Si
t2	t2.2xlarge	8	32	EBS solo	-	Moderada	Si
t3	t3.nano	2	0.5	EBS solo	Sí	Hasta 5 gigabits	Si
<input checked="" type="checkbox"/> t3	t3.micro	2	1	EBS solo	Sí	Hasta 5 gigabits	Si
t3	t3.small	2	2	EBS solo	Sí	Hasta 5 gigabits	Si
t3	Seleccionar T3.micro	2	4	EBS solo	Sí	Hasta 5 gigabits	Si
t3	t3.large	2	8	EBS solo	Sí	Hasta 5 gigabits	Si
t3	t3.xlarge	4	16	EBS solo	Sí	Hasta 5 gigabits	Si
t3	t3.2xlarge	8	32	EBS solo	Sí	Hasta 5 gigabits	Si

Hacer click en siguiente Cancelar Anterior Revisar y lanzar Siguiente: Página Configuración de los detalles de la instancia

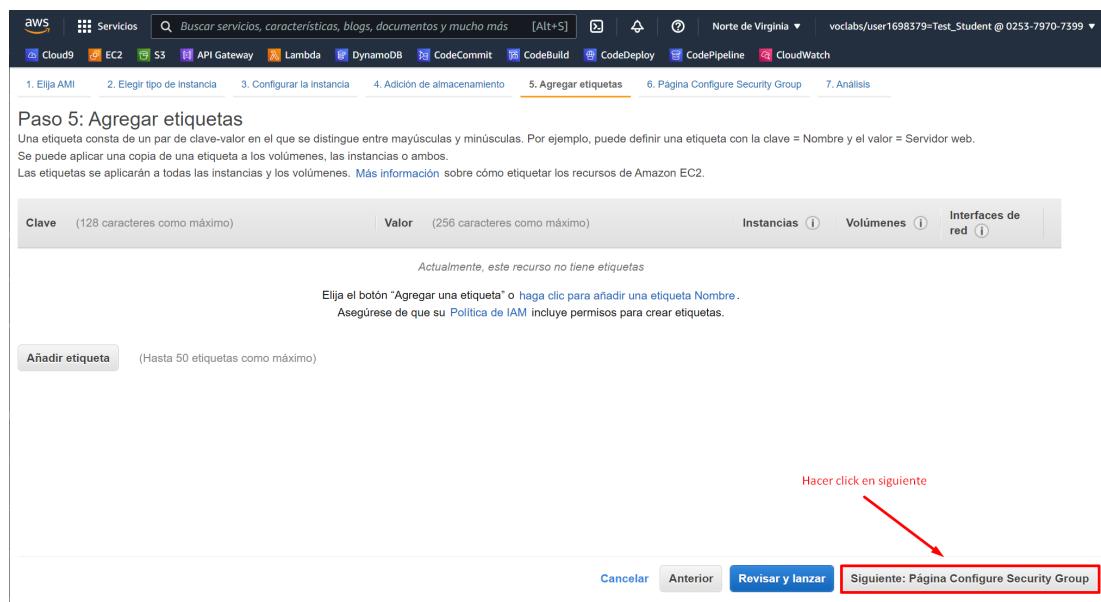
- En el paso 3, dejar todo como está salvo el rol de IAM de la instancia. Se ha de elegir el rol de **LabInstanceProfile**. A continuación, hacer clic en **Siguiente: Adición de almacenamiento**.



- En el paso 4, dejar todo como está, salvo el tamaño del volumen raíz. Se recomienda dar al menos **20GB** para evitar problemas posteriormente con el espacio en disco. Una vez modificado, hacer clic en **Siguiente: Agregar etiquetas**, para pasar a la siguiente ventana de configuración:



- En el paso 5, dejar todo como está y hacer clic en **Siguiente: página configure security groups**, para pasar a la siguiente ventana de configuración:



- Aquí se ha de crear un nuevo **Security Group** adaptado a las necesidades de del caso de uso, con el nombre de **labInstance**. Las reglas para definir son las siguientes:
- SSH con origen **personalizado**. El valor será de **0.0.0.0/0**, para que se pueda conectar desde el propio laboratorio, la máquina propia y el servicio de Cloud9. Se pueden aplicar permisos más restrictivos al servidor, siempre que el alumno sea consciente de los potenciales problemas ocasionados de ser muy estrictos.
 - HTTP con origen Mi IP, para dar acceso al puerto 80 de la instancia EC2.
 - HTTP con origen Mi IP, para dar acceso al puerto 8080 de la instancia EC2, para poder conectarse al servidor de Jenkins.

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 rules that allow unrestricted access to the 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: unir-server-sg
Description: unir-server-sg

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
Custom TCP	TCP	8080	My IP 92.191.56.253/32	e.g. SSH for Admin Desktop
HTTP	TCP	80	My IP 92.191.56.253/32	e.g. SSH for Admin Desktop

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.

Cancel Previous Review and Launch

- Antes de lanzar finalmente la instancia se ha de revisar la configuración final, para validar que está todo correcto y se lanza la instancia, haciendo clic en **Lanzar**.

Paso 7: Página Review Instance Launch

Revise los detalles de lanzamiento de su instancia. Retroceda para editar los cambios de cada sección. Haga clic en **Lanzar** para asignar un par de claves a la instancia y completar el proceso de lanzamiento.

⚠ La configuración de la instancia no cumple los requisitos de la capa de uso gratuita
Para lanzar una instancia que sea apta para la capa de uso gratuita, compruebe la selección de AMI, el tipo de instancia, las opciones de configuración o los dispositivos de almacenamiento. Obtenga más información sobre los requisitos y las restricciones de uso de [capa de uso gratuita](#).

Detalles de la AMI

Cloud9Ubuntu-2021-11-26T12-37 - ami-011bbfd2b87b54359
Cloud9 Cloud9Ubuntu AMI
Tipo de dispositivo raíz: ebs Tipo de virtualización: hvm

Tipo de instancia

Tipo de instancia	ECU	vCPU	Memoria (GiB)	Almacenamiento de la instancia (GB)	Optimizado para EBS disponible	Desempeño de la red
t3.micro	-	2	1	EBS solo	Sí	Up to 5 Gigabit

- Al momento de lanzar la instancia, se solicitará la utilización de claves de SSH para poder acceder a ella. Se seleccionará usar un par de claves existentes, eligiendo la clave llamada **vockey**. Finalmente se lanza la instancia.

Paso 7: Página Review Instance Launch

Revise los detalles de lanzamiento de su instancia. Retroceda para editar los cambios de cada sección. Haga clic en **Lanzar** para asignar un par de claves a la instancia y completar el proceso de lanzamiento.

Detalles de la AMI

Cloud9Ubuntu-2021-11-26T12-37 - ami-011bbfd2b87b54359

Tipo de instancia

Tipo de instancia	ECU	vCPU
t3.micro	1	2

Grupos de seguridad

Nombre del grupo de seguridad	Reglas de entrada	Descripción
SSH	TCP	
HTTP	TCP	
Regla TCP personalizada	TCP	

Selección de claves

Un par de claves consta de una clave pública que AWS almacena y un archivo de claves privadas que usted almacena. Juntas, le permiten conectarse a su instancia de forma segura. Para las AMI de Windows, el archivo de claves privadas es necesario para obtener la contraseña usada para iniciar sesión en la instancia. Para las AMI de Linux, el archivo de claves privadas le permite establecer una conexión SSH segura con su instancia. Amazon EC2 es compatible con los tipos de clave RSA y ED25519.

Notas: El par de claves seleccionado se añadirá al conjunto de claves autorizadas para esta instancia. Obtenga más información sobre [cómo eliminar pares de claves existentes de una AMI pública](#).

Opciones:

- Elegir un par de claves existente
- Seleccionar un par de claves
- vockey | RSA

Confirmación:

Confirmo que tengo acceso al archivo de clave privada correspondiente, y que si no existe este archivo, no podré iniciar sesión en mi instancia.

Botones:

Cancelar Lanzar instancias

Botones de navegación:

Cancelar Anterior Lanzar

- Una vez lanzada hay que revisar que se ha lanzado correctamente la instancia.



Página Launch Status

Se está lanzando su instancia
Se ha iniciado el siguiente lanzamiento de instancia: i-0c7e6798b2573207e [Ver log de lanzamiento](#)

Recibir notificaciones de los cargos estimados
Crear alertas de facturación para obtener una notificación por correo electrónico cuando los cargos estimados de su factura de AWS superen el importe definido (por ejemplo, cuando se excede la capa de uso gratuita).

Cómo conectarse a la instancia

Se está lanzando su instancia. Pueden transcurrir unos minutos hasta que tenga el estado **en ejecución**, momento en el cual estará lista para poder usarla. Las horas de uso de la nueva instancia comenzarán inmediatamente y seguirán devengando gastos hasta que detenga o termine la instancia.

Haga clic en [Ver las instancias](#) para monitorizar el estado de su instancia. Cuando la instancia tenga el estado **en ejecución**, podrá **conectarse** a ella desde la pantalla Instancias. [Más información](#) cómo conectarse a la instancia.

▼ Aquí tiene algunos recursos útiles que le ayudarán a comenzar

- [Cómo conectarse a la instancia Linux](#)
- [Amazon EC2: Guía del usuario](#)
- [Más información sobre la capa de uso gratuita de AWS](#)
- [Amazon EC2: Foro de debate](#)

Mientras se están lanzando sus instancias, también puede:

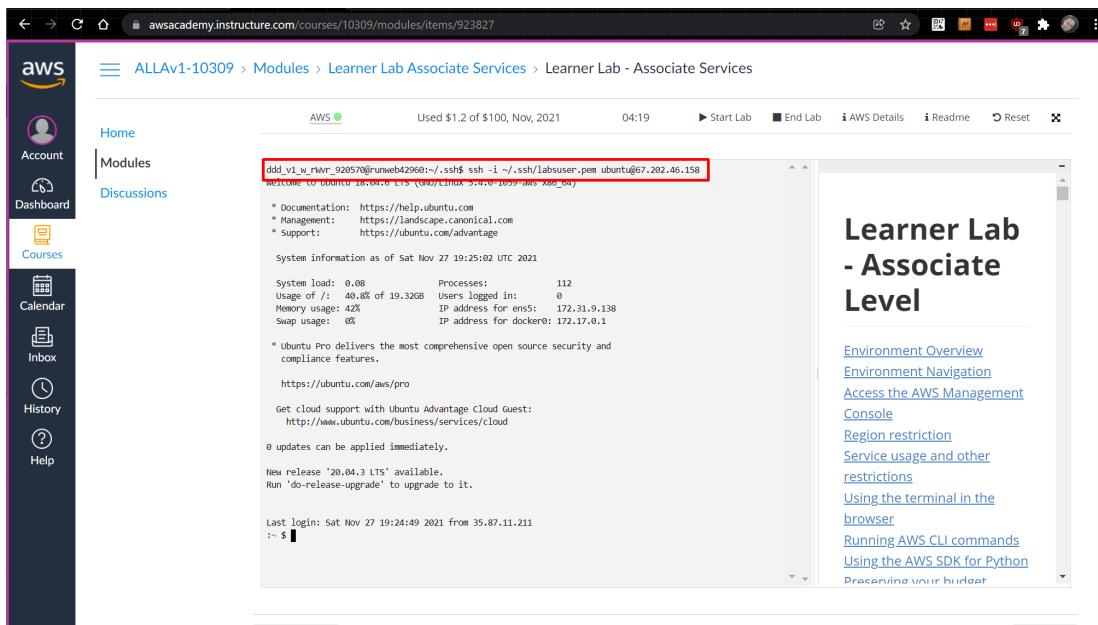
- [Crear alarmas de comprobación de estado](#) recibir notificaciones cuando estas instancias no superen las comprobaciones de estado. (Podrían aplicarse cargos adicionales)
- [Crear y asociar volúmenes de EBS adicionales](#) (Podrían aplicarse cargos adicionales)
- [Administrar grupos de seguridad](#)

[Ver instancias](#)

Al hacer clic en el botón **Ver instancias** se podrá acceder a la información de la instancia, como puede ser su FQDN público, su IP pública, su IP privada, el almacenamiento disponible, los Security Groups asociados, etc...

- Por último, iniciar sesión como el usuario **ubuntu**. Por ejemplo, ejecutar este comando en el terminal del laboratorio, donde public-ip es la dirección IP pública real:

```
ssh -i ~/.ssh/labsuser.pem ubuntu@public-ip
```



Integración con Cloud9

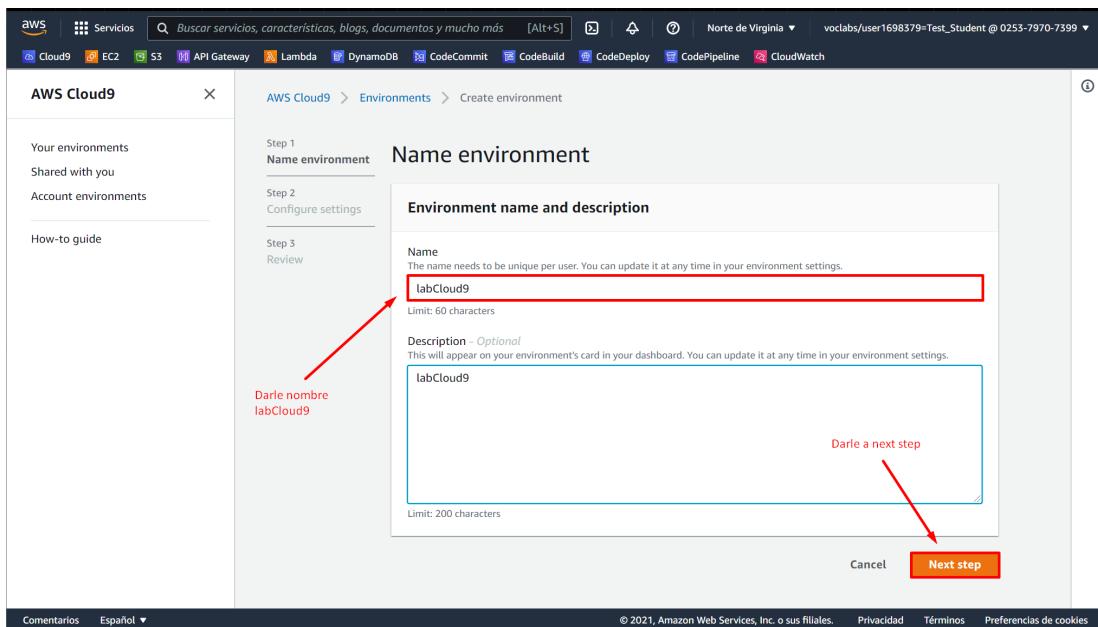
A continuación, se va a integrar el servidor EC2 levantado, con el servicio de Cloud9, para poder trabajar en un IDE más cómodo que el terminal de Linux que puede ofrecer AWS Academy. Para ello hay que ir al servicio de Cloud9 y seguir los siguientes pasos:

- Crear un nuevo entorno, haciendo clic en el botón **Create Environment**.

The screenshot shows the AWS Cloud9 landing page. At the top, there's a navigation bar with the AWS logo, a search bar, and various service links like Cloud9, EC2, S3, API Gateway, Lambda, DynamoDB, CodeCommit, CodeBuild, CodeDeploy, CodePipeline, and CloudWatch. The main heading is "AWS Cloud9: A cloud IDE for writing, running, and debugging code". Below the heading, there's a brief description of what AWS Cloud9 is and how it works. A prominent red button labeled "Create environment" is highlighted with a red arrow pointing to it. To the right of the button, there's a tooltip in Spanish: "Crear nuevo entorno de Cloud9". On the left, there's a section titled "How it works" with a "Learn more" link. On the right, there's a "Getting started" sidebar with links to "Before you start", "Create an environment", "Working with environments", "Working with the IDE", and "Working with AWS Lambda", each with a corresponding "min read" time. Below that is a "More resources" sidebar with links to "FAQs", "Forum", and "Contact us".

- Generar el entorno asignándole nombre, descripción y haciendo clic sobre la opción **Next Step**. Los valores serán los siguientes:

- **Name:** labCloud9
- **Description:** labCloud9



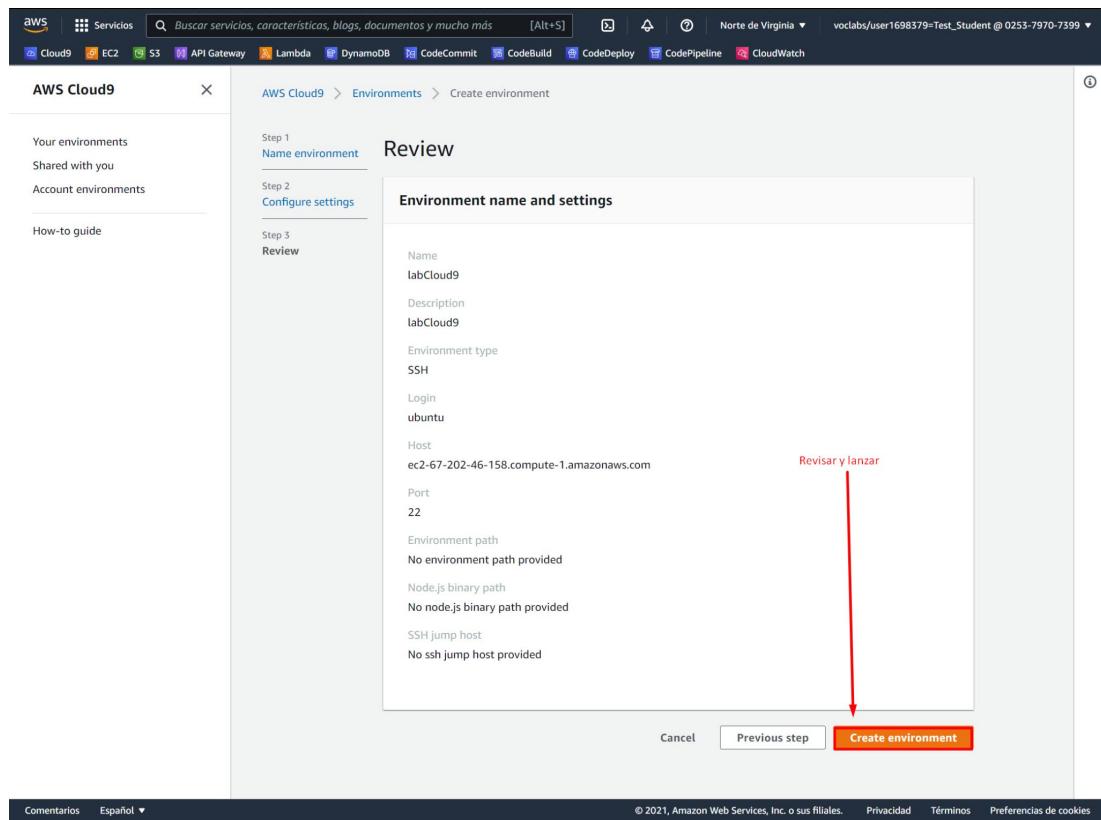
- Aquí es donde se va a integrar el servicio de Cloud9 con la instancia EC2 que se ha levantado previamente. Es especialmente importante prestar atención a la configuración para evitar errores. Los parámetros que se han de definir son:
 - Create and run in remote server (SSH connection)
 - User: Ubuntu
 - Host: hostname público del servidor EC2 que se ha levantado previamente
 - Port: 22
 - Copy key to clipboard. El valor copiado se ha de pegar en el fichero authorized_keys de la instancia EC2, para que el servicio de Cloud9 pueda configurar el entorno. **No continuar con la configuración hasta no haber copiado el valor en el servidor.**

```
vim ~/.ssh/authorized_keys
```

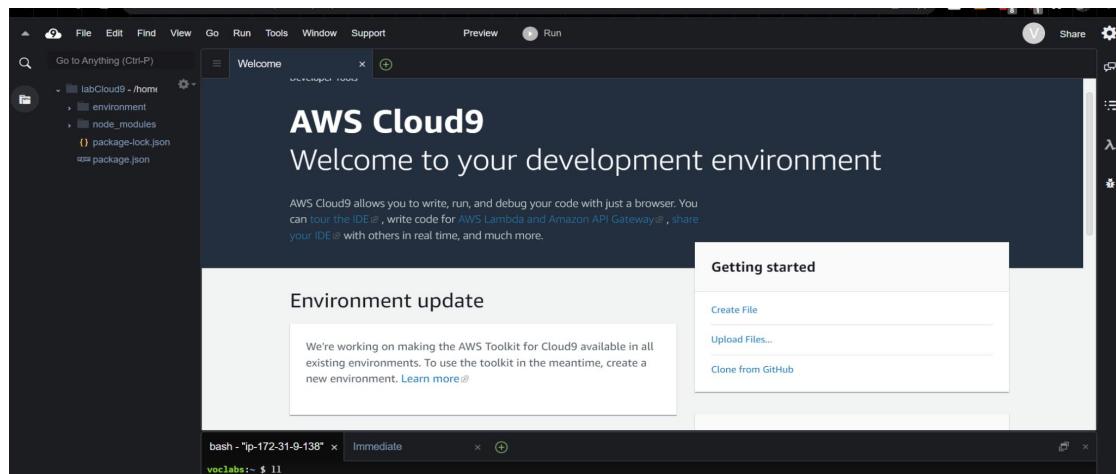
The screenshot shows the 'Configure settings' step of the AWS Cloud9 environment creation process. The 'Environment type' is set to 'Create a new EC2 instance for environment (direct access)'. The 'User' field contains 'ubuntu', 'Host' contains 'ec2-67-202-46-158.compute-1.amazonaws.com', and 'Port' is set to '22'. A red box highlights the 'Copy key to clipboard' button, with a tooltip 'Copiar para pegarlo dentro del authorized_keys en el servidor EC2'. Below the host field, there's a note: 'AWS Cloud9 can use an SSH public key to connect securely to your server. To start, you need to add our public key to your ~/.ssh/authorized_keys file and provide your remote login credentials below.' At the bottom, there are 'Cancel', 'Previous step', and a red-highlighted 'Next step' button.

Una vez copiado el valor de la clave pública de SSH en el servidor, continuar con la configuración de Cloud9, dando al botón **Next Step**.

- ▶ Finalmente, revisar que la configuración para validar que todos los parámetros son correctos y hacer clic en **Create Environment**.



- ▶ Finalmente, si todo va bien, se redirigirá automáticamente al entorno de desarrollo de Cloud9:



Una vez integrado Cloud9 en el servidor EC2, se procede a la instalación de Jenkins.

Integración del servicio de Jenkins

A continuación, se han de seguir estos pasos para instalar la instancia de Jenkins en el servidor EC2. Para ello se ha de conectar al servidor EC2, bien a través del propio laboratorio, o desde el equipo personal, usando la clave **vockey**, mencionada anteriormente.

Instalación del servidor de Jenkins

Primero se ha de configurar Java:

- ▶ Configuración de Java en Ubuntu:

```
# Actualizar listado de paquetes del sistema
$ sudo apt-get update
```

- ▶ Instalar Java 11:

```
# Instalar default-jdk
$ sudo apt-get install default-jdk -y
```

```
Reading package lists... Done
vclabs:~ $ sudo apt-get install default-jdk -y
Reading package lists... Done
Building dependency tree
Reading state information... done
The following additional packages will be installed:
at-sp12-core default-jdk-headless default-jre default-jre-headless fonts-dejavu-extra libatk-bridge2.0-0 libatk-wrapper-java libatk-wrapper-java-jni
libatk1.0-0 libatk1.0-data libatspi2.0-0 libdrm-amdgpu libdrm-intel1 libdrm-nouveau1 libdrm-radeon1 libfontenc1 libgif7 libgl1 libgl1-mesa-dri
libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6 libl1vmt0 libpciaccess0 libpthread-stubs0-dev libsensors4 libsm-dev libsm6 libxi1-dev
libxi1-doc libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-shape0 libxcb-sync1 libxcb1-dev
libxcomposite1 libxdamage1 libxdmcp-dev libxfixes3 libxf86_drm libxi6 libxinerama1 libxmu6 libxpm4 libxrandr2 libxrender1 libxshmfence1 libxt-dev
libxtst6 libxv1 libxf86dg1 libxf86vm1 openjdk-11-jdk openjdk-11-jre x11-common x11-utils x11proto-core-dev x11proto-dev xorg-sgml-doctools
xtrans-dev
Suggested packages:
libice-doc lm-sensors libsm-doc libxcb-doc libxt-doc openjdk-11-demo openjdk-11-source visualvm mesa-utils
The following NEW packages will be installed:
at-sp12-core default-jdk default-jdk-headless default-jre default-jre-headless fonts-dejavu-extra libatk-bridge2.0-0 libatk-wrapper-java
libatk-wrapper-java-jni libatk1.0-0 libatk1.0-data libatspi2.0-0 libdrm-amdgpu libdrm-intel1 libdrm-nouveau1 libdrm-radeon1 libfontenc1 libgif7
libgl1 libgl1-mesa-dri libglapi-mesa libglvnd0 libglx-mesa0 libglx0 libice-dev libice6 libl1vmt0 libpciaccess0 libpthread-stubs0-dev libsensors4
libsm-dev libsm6 libxi1-dev libxi1-doc libx11-xcb1 libxau-dev libxaw7 libxcb-dri2-0 libxcb-dri3-0 libxcb-glx0 libxcb-present0 libxcb-shape0
libxcb-sync1 libxcb1-dev libxcomposite1 libxdamage1 libxdmcp-dev libxfixes3 libxf86_drm libxi6 libxinerama1 libxmu6 libxpm4 libxrandr2 libxrender1
libxshmfence1 libxt-dev libxt6 libxv1 libxf86dg1 libxf86vm1 openjdk-11-jdk openjdk-11-jre x11-common x11-utils x11proto-core-dev
```

A continuación, se preparan los repositorios para la instalación de Jenkins:

- ▶ Configuración de Jenkins Agregar clave de repositorio al sistema:

```
# Agregar claves del repositorio de Jenkins
$ wget -qO - https://pkg.jenkins.io/debian-stable/jenkins.io.key | 
sudo apt-key add -
```

- ▶ Agregar la dirección de repositorio del paquete debían al sistema:

```
# Agregar repositorio a los source.list de Ubuntu
$ echo deb http://pkg.jenkins.io/debian-stable binary/ | sudo tee
/etc/apt/sources.list.d/jenkins.list
```

- ▶ Actualizar el listado de paquetes del sistema:

```
# Actualizar listado de paquetes del sistema
$ sudo apt-get update
```

Instalación del paquete de Jenkins:

- ▶ Instalación del paquete:

```
# Instalar Jenkins server
$ sudo apt-get install jenkins -y
```

```

ubuntu@ip-172-31-37-211:~$ sudo apt-get install jenkins -y
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  daemon
The following NEW packages will be installed:
  daemon jenkins
0 upgraded, 2 newly installed, 0 to remove and 8 not upgraded.
Need to get 72.0 MB of archives.
After this operation, 75.1 MB of additional disk space will be used.
Get:1 http://us-east-2.ec2.archive.ubuntu.com/ubuntu xenial/universe amd64 daemon
Get:2 http://pkg.jenkins.io/debian-stable binary/ jenkins 2.121.3 [71.9 MB]
Fetched 72.0 MB in 5s (12.5 MB/s)
Selecting previously unselected package daemon.
(Reading database ... 51843 files and directories currently installed.)
Preparing to unpack .../daemon_0.6.4-1_amd64.deb ...
Unpacking daemon (0.6.4-1) ...
Selecting previously unselected package jenkins.
Preparing to unpack .../jenkins_2.121.3_all.deb ...
Unpacking jenkins (2.121.3) ...
Processing triggers for man-db (2.7.5-1) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for ureadahead (0.100.0-19) ...
Setting up daemon (0.6.4-1) ...
Setting up jenkins (2.121.3) ...
Processing triggers for systemd (229-4ubuntu21.4) ...
Processing triggers for ureadahead (0.100.0-19) ...
ubuntu@ip-172-31-37-211:~$ 

```

- ▶ La captura de pantalla anterior debería confirmar que Jenkins se instaló correctamente. También se puede validar que el servicio está ejecutando, a través del siguiente comando:

```

# Verificar estado servidor de Jenkins
$ sudo service jenkins status

```

```

voclabs:~ $ sudo service jenkins status
● jenkins.service - Jenkins Continuous Integration Server
  Loaded: loaded (/lib/systemd/system/jenkins.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-04-28 17:47:03 UTC; 35s ago
    Main PID: 8675 (java)
       Tasks: 41 (limit: 1104)
      CGroup: /system.slice/jenkins.service
              └─8675 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Apr 28 17:46:32 ip-172-31-81-101 jenkins[8675]: Please use the following password to proceed to installation:
Apr 28 17:46:32 ip-172-31-81-101 jenkins[8675]: 665a3dfb107a4f1f983a7fa943c71698
Apr 28 17:46:32 ip-172-31-81-101 jenkins[8675]: this may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Apr 28 17:46:32 ip-172-31-81-101 jenkins[8675]: *****
Path de la contraseña inicial del Administrador
Apr 28 17:47:03 ip-172-31-81-101 jenkins[8675]: 2022-04-28 17:47:03.122+0000 [id=38]           INFO    jenkins.InitReactorRunner$1@onAttained: Completed
Apr 28 17:47:03 ip-172-31-81-101 jenkins[8675]: 2022-04-28 17:47:03.150+0000 [id=22]           INFO    hudson.lifecycle.lifecycle#onReady: Jenkins is fu
Apr 28 17:47:03 ip-172-31-81-101 systemd[1]: Started Jenkins Continuous Integration Server.
Apr 28 17:47:03 ip-172-31-81-101 jenkins[8675]: 2022-04-28 17:47:03.325+0000 [id=46]           INFO    h.m.DownloadService$Downloadable#load: obtained t
Apr 28 17:47:03 ip-172-31-81-101 jenkins[8675]: 2022-04-28 17:47:03.326+0000 [id=46]           INFO    hudson.util.Retrier#start: Performed the action c
Apr 28 17:47:03 ip-172-31-81-101 jenkins[8675]: 2022-04-28 17:47:03.332+0000 [id=46]           INFO    hudson.model.AsyncPeriodicWork$#doRun$1: Fi
lines 1-18/18 (END)

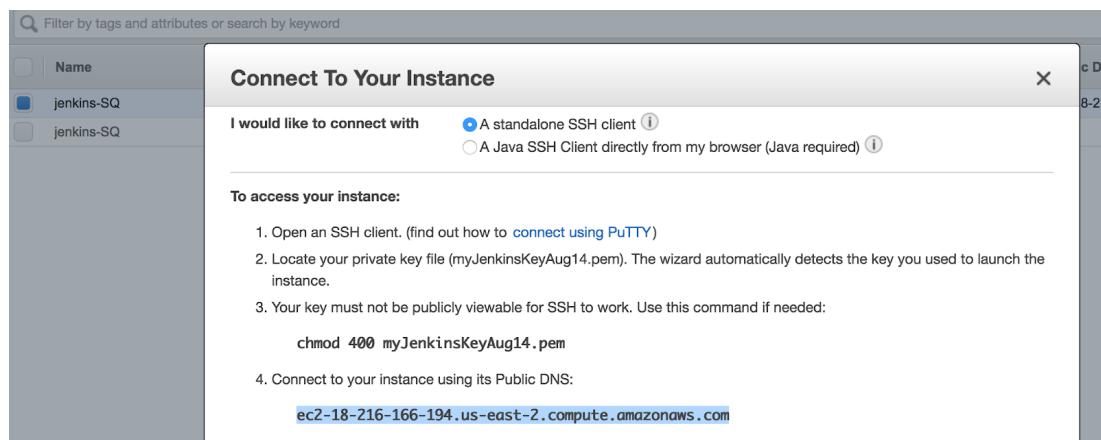
```

Configuración de Jenkins

Una vez instalado Jenkins, se ha de hacer una configuración básica, configurando sus credenciales de administración y los plugins que se van a instalar. Los pasos son:

Acceder al Jenkins a través de la URL:

- ▶ Ir a la consola de EC2 en la AWS.
- ▶ Hacer clic en EC2, y después clic en el enlace de instancias en ejecución.
- Seleccionar la casilla de verificación de EC2 que está instalando Java y Jenkins.
- ▶ Hacer clic en Acción.
- ▶ Copiar el valor del paso 4 que dice -> **Conéctese a su instancia usando su DNS público:**



- ▶ Ahora hay que ir al navegador. Ingresar el nombre de DNS público o la dirección IP pública con el puerto 8080. Así es como seleccionar el nombre de DNS público:

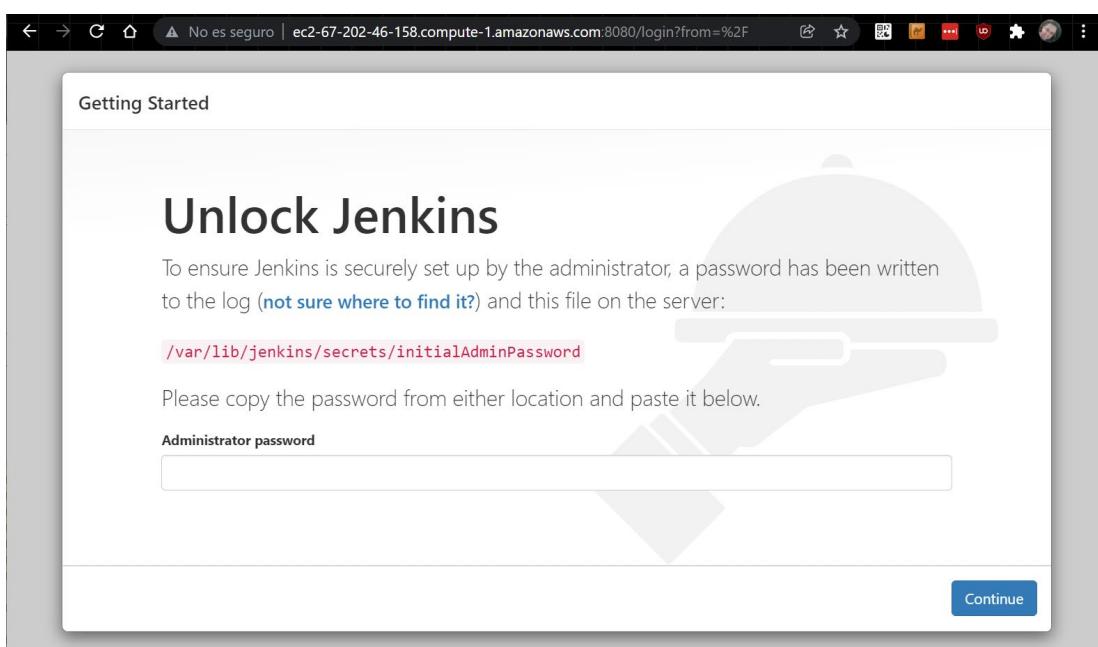
```
# URL pública del servidor de Jenkins
$ http://$EC2_public_dns_name:8080
```

Desbloquear Jenkins

- ▶ A continuación, se ha de desbloquear Jenkins con el valor que se nos solicita en la web de Jenkins. Para ello hay que ejecutar el siguiente comando desde la instancia

EC2 para poder copiar la clave para continuar y pegarla en el navegador. Una vez pegada hacer clic en **Continue**:

```
# Cat del fichero con el password inicial de Jenkins  
$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
```



- ▶ Seguidamente se han de instalar los plugins de Jenkins. Se recomienda instalar los plugins sugeridos por Jenkins. Por tanto, hacer clic en instalar complementos sugeridos.

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

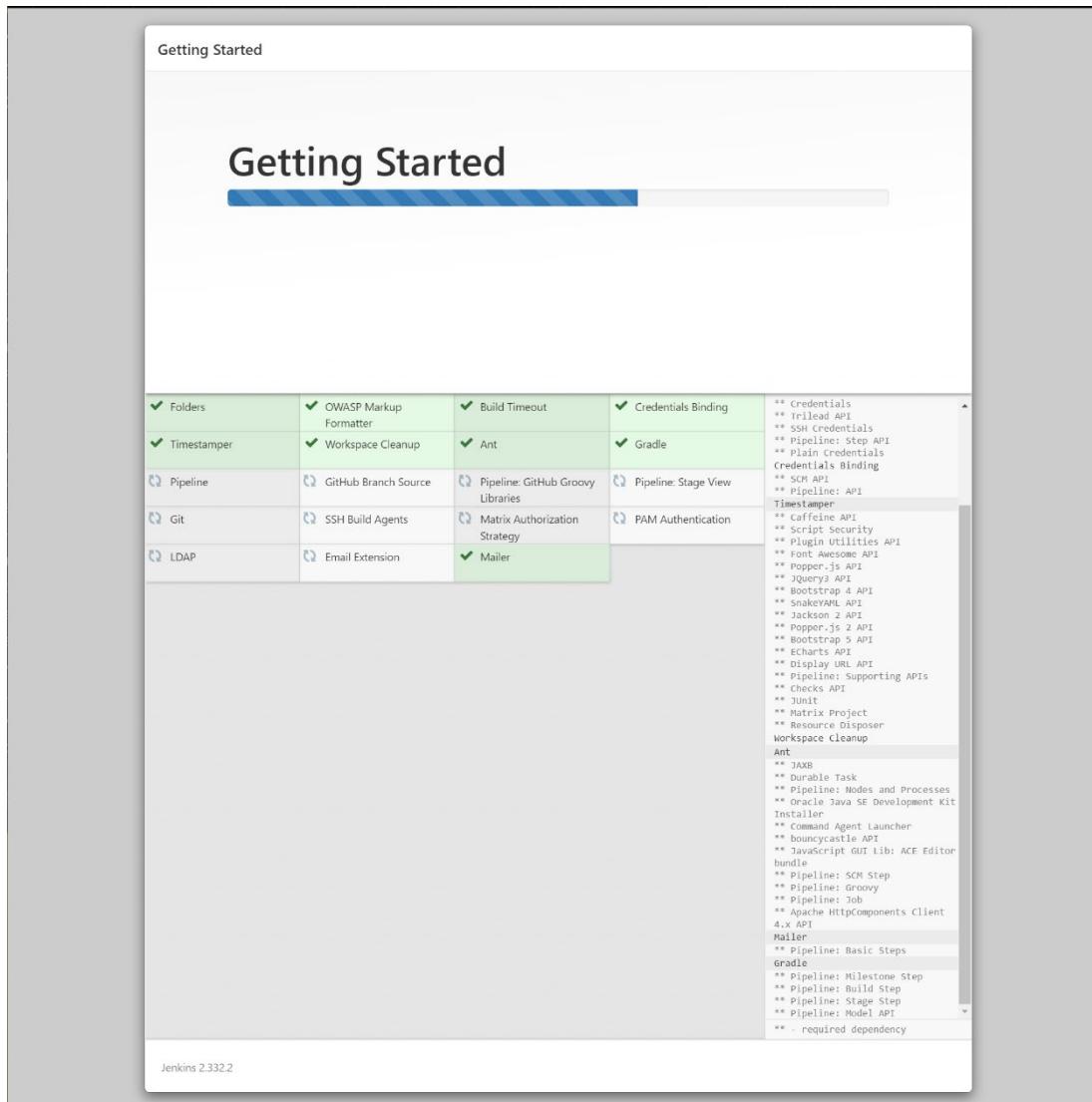
Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

- ▶ A continuación se lleva la instalación de automática de plugins por defecto en el servidor de integración continua Jenkins, tal como apreciarse en la siguiente captura de pantalla:



- ▶ Una vez instalados los plugin, se continúa con el proceso. Por tanto, se ha de crear el primer administrador de Jenkins, usando para ello los valores que decida el alumno. Se hace clic en **Save and Continue** para pasar a la siguiente pantalla de configuración.

Getting Started

Create First Admin User

Usuario:

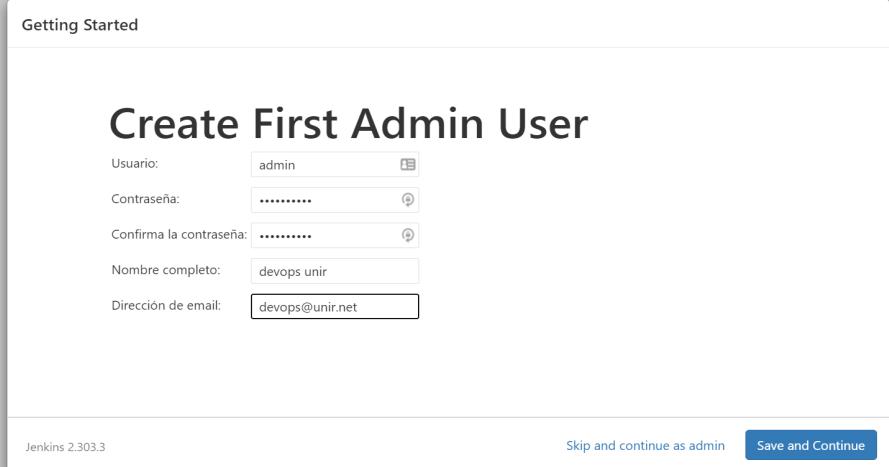
Contraseña:

Confirma la contraseña:

Nombre completo:

Dirección de email:

Jenkins 2.303.3 Skip and continue as admin Save and Continue



- ▶ En el siguiente paso se solicitará la URL del servidor donde esté Jenkins instalado. Este punto se puede saltar, indicando **Not Now**, o bien se puede configurar, indicando el FQDN de la instancia EC2. En caso de incluir el FQDN, hay que tener presente que, si se reinicia el servidor y cambia la IP pública de Jenkins, el FQDN también cambiará y Jenkins dará una advertencia al detectar un problema de configuración. Hacer clic en **Save and Finish** para finalizar.

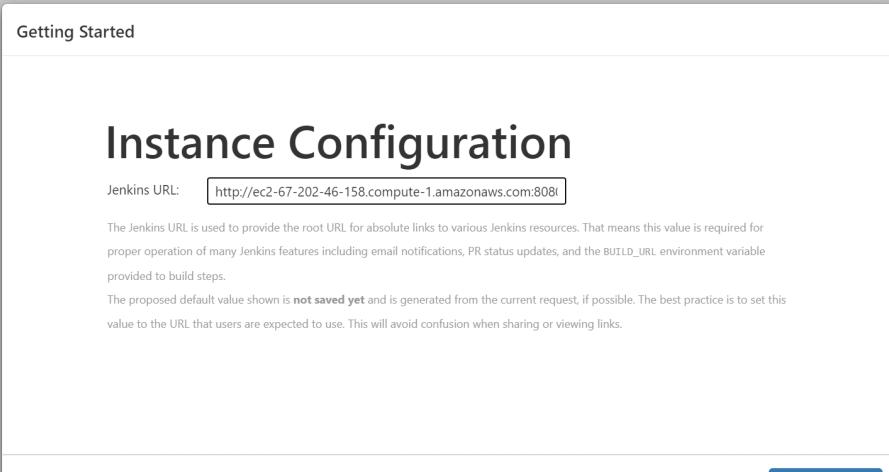
Getting Started

Instance Configuration

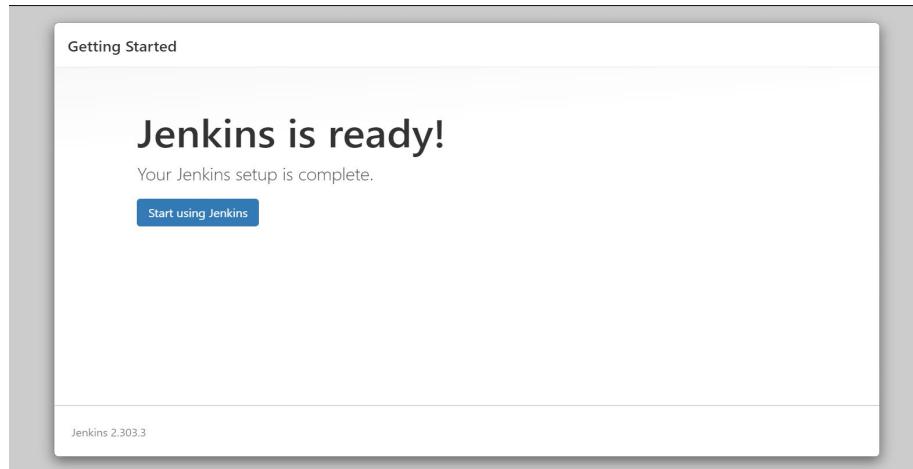
Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.303.3 Not now Save and Finish



- ▶ Finalmente, el servidor de Jenkins debería de estar preparado. Por tanto, hay que hacer clic en **Start using Jenkins** para trabajar en Jenkins.



- ▶ El alumno deberá de acceder ya al servidor de Jenkins normalmente para el desarrollo normal de la práctica:

Instalación de plugins adicionales

El alumno también deberá de instalar una serie de plugins adicionales, como son el plugin de Coverage, para que se reporten las pruebas de cobertura en el propio Jenkins y el plugin de SSH agent. Para ello hay que seguir los siguientes pasos:

- Hacer clic en Administrar Jenkins:

The screenshot shows the Jenkins dashboard. On the left sidebar, there is a link labeled "Administrar Jenkins". A red arrow points from the text "Click en administrar Jenkins" at the bottom right of the sidebar towards this link. The main content area displays a table of pipelines: PIPELINE-FULL-CD, PIPELINE-FULL-PRODUCTION, and PIPELINE-FULL-STAGING. At the bottom right of the dashboard, there are links for REST API and Jenkins 2.319.1.

- Una vez en la ventana de Administrar Jenkins, hacer clic en **Administrar plugins**:

The screenshot shows the "System Configuration" page under "Estado del ejecutor de construcciones". On the right side, there is a section titled "Administrador de Jenkins" which includes a link "Administrar Plugins". A red arrow points from the text "Clic en administrar plugins" at the bottom right of this section towards this link. The page also contains sections for "Configuración del sistema", "Configuración global de la seguridad", "Gestión de usuarios", and "Status Information".

- ▶ Finalmente, se han de instalar librerías, utilizando el gestor de plugins, haciendo lo siguiente:
 - Seleccionar la pestaña **Todos los plugins**.
 - En la búsqueda, introducir **Cobertura** y marcar para instalar.
 - Repetir, buscando en esta ocasión **SSH agent**.
 - Una vez marcados ambos paquetes, hacer clic en **Install without restart**.

The screenshot shows the Jenkins Plugin Manager interface. At the top, there's a search bar with the text "ssh-agent". Below it, a navigation bar has the "Todos los plugins" tab selected. A red arrow points to this tab. The main area lists two plugins:

Name	Version	Released
Cobertura	1.17	Hace 1 Mes 2 días
SSH Agent	1.23	Hace 5 Mes 29 días

For each plugin, there are "Install" and "Download now and install after restart" buttons. A red arrow points to the "Install" button for the SSH Agent plugin. At the bottom of the page, there are links for "REST API" and "Jenkins 2.319.1".

- Verificar que la instalación es exitosa y volver al inicio de la página.



Instalación de librerías de Python

Puesto que la práctica se encuentra desarrollada en Python3.7, será necesario instalar tanto Python3.7, como la librería para poder crear entornos virtuales en Python3.7 en la instancia EC2.

```
# Instalación python3.7 y virtualenv de python3.7  
$ sudo apt install python3.7 python3.7-venv
```

Introducción de la práctica

Formas parte de un equipo multidisciplinar, siendo responsable de la implantación de todo ciclo de operativización de desarrollos que parten del pseudocódigo hasta el entorno productivo. Como DevOps especializado en la plataforma tecnológica Cloud de AWS. Un día un cliente te pide que le ayudes a productivizar una aplicación que está desarrollando, especialmente la parte de Backend, porque ha montado todo de manera manual y cada vez que realiza un cambio observa que pierde mucho tiempo haciendo pruebas y actualizando todos los servicios, puesto que no tiene ningún tipo de *pipeline* de CI/CD ni pruebas unitarias ni de integración. Esta aplicación se trata de una API RESTful de libreta de tareas pendientes (ToDo). Para ello, te propone el uso de la tecnología Serverless Application Model (SAM) y Jenkins:

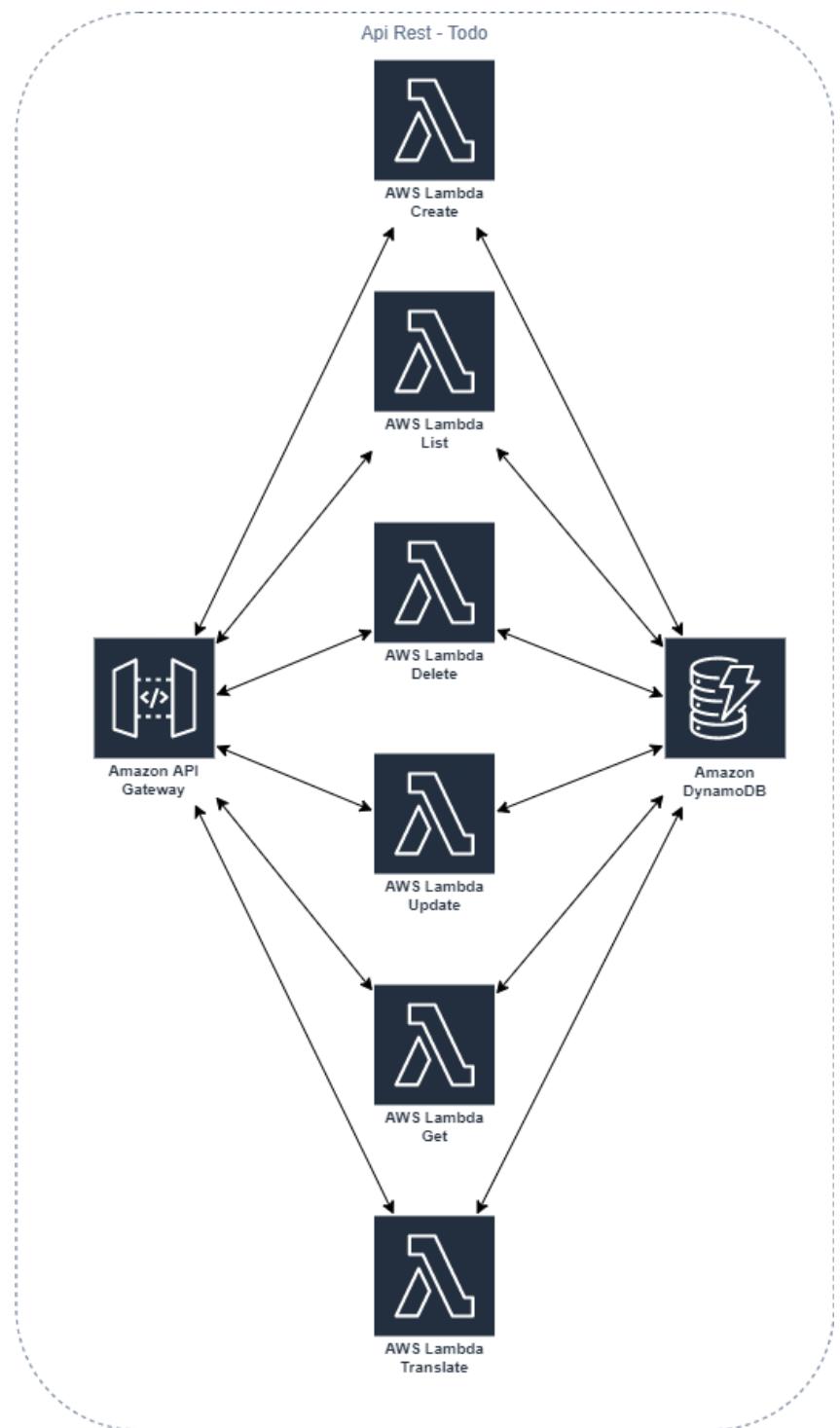
- ▶ [SAM](#) + [Jenkins](#): El modelo de aplicaciones sin servidor (SAM) de AWS es un framework open source pensado para construir aplicaciones sin servidor. Proporciona una sintaxis abreviada para expresar funciones, API, bases de datos y mapeos de fuentes de eventos. Con sólo unas pocas líneas por recurso, se puede definir la aplicación que se desee y modelar usando YAML. Durante la implementación, SAM transforma y expande la sintaxis de SAM en la sintaxis de AWS CloudFormation, permitiéndole construir aplicaciones sin servidor más rápidamente. Por otro lado, Jenkins es un servicio de automatización de procesos de desarrollo software que facilita determinadas tareas de la integración continua y de la entrega continua.

Adicionalmente, quiere que amplíes la funcionalidad base, incorporando una nueva *feature*. Esta *feature* corresponde a una función nueva que sea capaz de traducir un registro único de la lista de ToDos, al idioma que se le solicite a través del API. Para ello te da la libertad de usar servicios autogestionados de AWS, como puedan ser AWS Comprehend o AWS Translate.

El trabajo será a partir de un *software* base, y necesita tu valoración acerca de la mejor solución posible, y además ampliar el aplicativo, robusteciendo el mismo con baterías de pruebas que certifiquen la calidad SW entregado al cliente. El código base de la práctica está basado en los ejemplos del framework serverless de ToDo list, disponible [aquí](#).

El software base o de partida lo constituye un *Backend* de servicios, resueltos como implementación API REST que aprovisiona una aplicación de listado de tareas (formalmente conocido como [ToDo list](#)), cuya arquitectura está compuesta por los siguientes servicios:

- ▶ [Amazon API Gateway](#): Vía de creación, publicación, el mantenimiento, monitorización y protección de API de servicios empresariales con comunicación bidireccional y gestión de procesamiento de miles de llamadas simultáneas a los métodos de servicio disponibles.
- ▶ [AWS Lambda](#): Canal de ejecución de código sin requerimiento de aprovisionamiento de servidores ni configuración previa de escalado según necesidades en la carga de trabajo demandada.
- ▶ [AWS DynamoDB](#): Base de datos autogestionada de tipo clave-valor documental, y alto rendimiento transaccional. Sistema de almacenamiento y consulta de las tareas por parte del *Backend* de servicios.
- ▶ [AWS Cloudwatch](#): Ubicación de publicación de indicios (trazas, métricas, alarmas programadas) acerca del estado de funcionamiento de cada uno de los métodos de servicio requeridos para la correcta disponibilidad del *Backend* que aprovisiona la aplicación de ToDo-list.



Previamente a la explicación de cada uno de los apartados que componen el caso práctico 1, se indica una tabla comparativa de comparación de servicios a requerir que puedan servir de guía comparativa a futuro para el alumno:

Etapa de <i>Pipeline</i>	Apartado A
Control de código	Github
Orquestación CICD	Jenkins
Build	Jenkins + SAM
Unit Test	unittest o PyTest
Coverage Test	coverage
Quality Test	flake8
Security Test	bandit
Complexity Test	radon · PyPI
Deployment Infra	SAM + Jenkins
Lógica de negocio	AWS Lambda
Persistencia (Backend)	DynamoDB
API	API Gateway
Monitorización	AWS Cloudwatch

Caso práctico 1. Apartado A

En línea con lo aprendido durante el programa, AWS dispone de una suite de servicios orientados a como desplegar aplicaciones desde cero a través de *Serverless Application Model (SAM)*. Su uso posibilita la construcción de *pipelines* de integración y entrega continua para automatizar los procesos de compilación de los artefactos software requeridos en el despliegue en el entorno productivo.

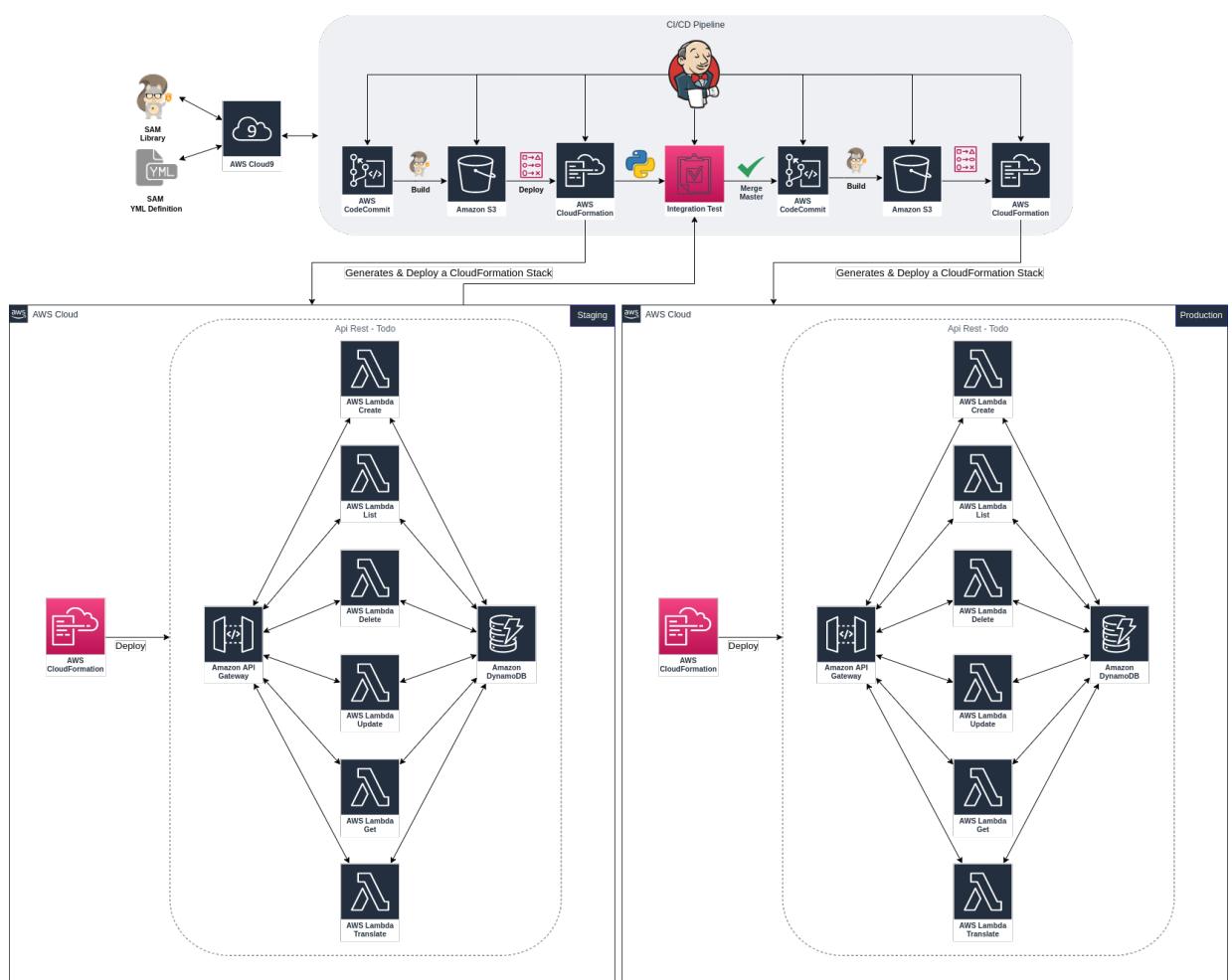
A continuación, se van a detallar brevemente cada uno de los servicios necesarios para la elaboración de este apartado:

- ▶ [AWS CloudFormation](#): Servicio de diseño, implementación y despliegue automático de infraestructura de aplicaciones Cloud, simplificando su diseño a través de un lenguaje común de modelado.
- ▶ [Elastic Computer Service \(EC2\)](#): Servicio de computación en la nube de AWS, según la cual el usuario es capaz de disponer al instante, y sin inversión previa en infraestructura hardware física propia, de capacidad informática acorde a las necesidades de las aplicaciones o soluciones digitales. El modelo de negocio y explotación se rige según la demanda a cada instante de la empresa u organización en el uso de dicho servicio en cuestión, optimizando ostensiblemente los costes asociados por ello.
- ▶ [Simple Storage Service \(S3\)](#): Servicio de almacenamiento de objetos con sistema de versionado ante modificaciones en los mismos, alto rendimiento y finalidad multipropósito (Lago de datos y tracking IoT, sitios web, aplicaciones mobile, recurso de backup y archivado, entre otros).

Además de los servicios de AWS, se va a hacer uso de la herramienta de [Jenkins](#), desplegada dentro de una instancia EC2 para construir ahí los diferentes *pipelines* que se van a proponer en este apartado.

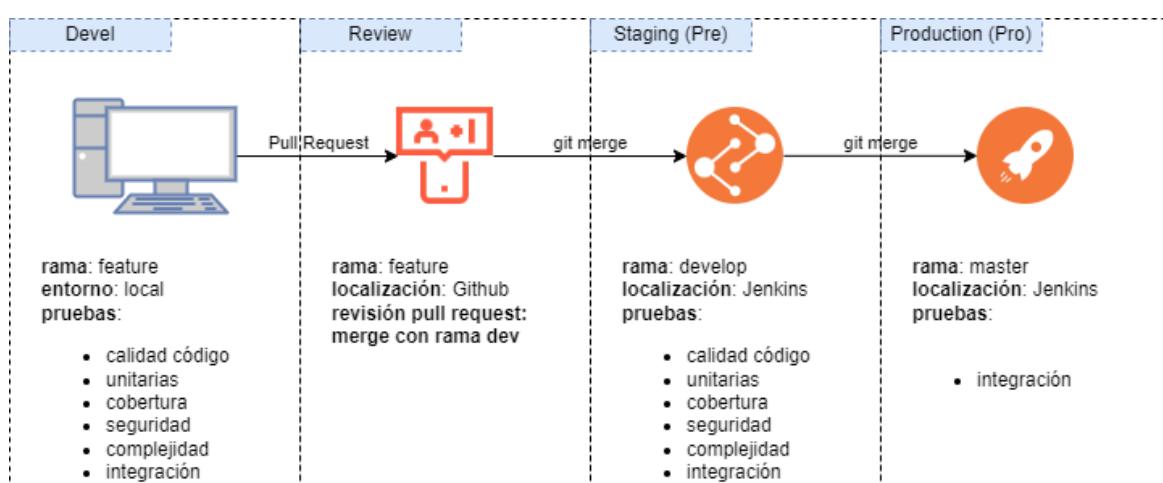
Resumen de la solución. Apartado A

El objetivo global es el de adquirir un conocimiento extenso en el uso de distintos *frameworks* de diseño e implementación de *pipelines* de CI/CD, siendo esta aproximación la correspondiente al ámbito de trabajo del propio proveedor de soluciones en la nube Amazon Web Services, utilizando el servicio más común en los entornos productivos, como es Jenkins y el marco de despliegue de arquitectura de aplicaciones Software Serverless AWS SAM. La visión global que el alumno debe de lograr alcanzar de la solución en esta siguiente ocasión ha de ser similar al siguiente:



Para ello se propone un *Pipeline* de CI/CD basado en 2 entornos físicos:

- ▶ Local: donde se desarrollarán y probarán las nuevas features desplegadas de manera local. Requerirá de usar una rama nueva de git denominada ***new-feature-one***.
- ▶ CI/CD: con dos escenarios de preproducción (*staging*) y producción (*production*), donde se construirá el *Pipeline* de CI/CD. En cada escenario se usará la rama adecuada para cada entorno, siendo ***develop*** para el entorno de preproducción y ***master*** para el entorno de producción. En este segundo ejercicio el alumno tiene que implementar las distintas pruebas de sobre el código de manera obligatoria.



De cara a la elaboración de apartado A, deberán afrontarse las siguientes fases o etapas desde la cuenta asignada a cada alumno en AWS Academy, de las que se entrará en mayor detalle seguidamente:

- 1. Clonado repositorio de la práctica y copia en repositorio de alumno**
- 2. Validación SAM CLI (Command Line Interface) y análisis de repositorio**
- 3. Ejecución de proyecto en entorno local (SAM CLI)**
- 4. Despliegue manual de aplicación SAM en AWS**
- 5. Configuración entorno Jenkins**
- 6. Creación de *pipelines* de Jenkins para despliegue de arquitectura completa**
 - 6.1. Pipeline de Staging**
 - 6.2. Pipeline de Production**
 - 6.3. CI/CD completo**

Clonado repositorio de la práctica y copia en repositorio de alumno

Configuración inicial

1. Crear una cuenta en [Github](#). El alumno deberá de crearse una cuenta en Github, para almacenar los repositorios de código de la práctica. **No se debe de trabajar sobre el repositorio de los profesores de la práctica de ninguna de las formas.**
2. Crear un repositorio llamado **todo-list-aws** en la cuenta del alumno, siguiendo la [guía oficial para crear un repositorio de Github](#)
3. Conectarse al servidor EC2, integrado con Cloud9 y Jenkins mediante SSH y clonar el repositorio de la práctica, alojado en:
<https://github.com/rgaleanog/todo-list-aws.git>

```
# Clonado del repositorio original
$ git clone https://github.com/rgaleanog/todo-list-aws.git todo-list-aws

# Directorio del proyecto
$ cd todo-list-aws/
```

4. Una vez clonado, cambiar, cambiar los orígenes del repositorio en el servidor, para que apunten al nuevo repositorio en lugar del repositorio original de Github. Es importante verificar primero la ruta del repositorio del alumno en Github antes, para no ejecutar el comando y apuntar a otra URL o a una URL que no exista:

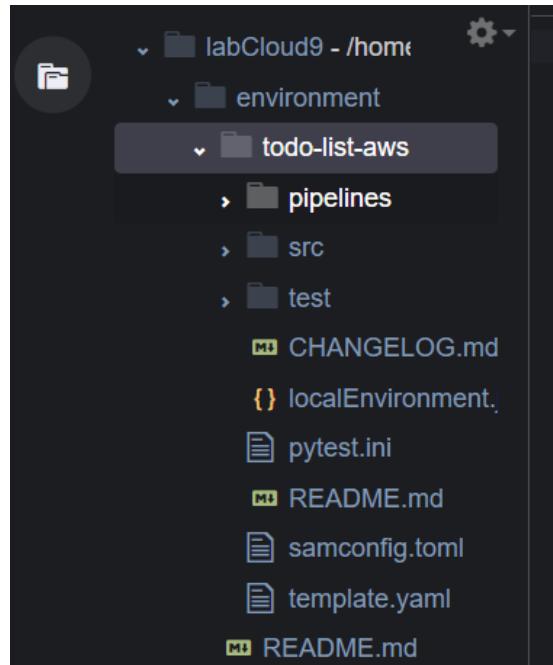
```
git remote -v
# Ver remotes actuales
# origin  https://github.com/rgaleanog/todo-list-aws.git (fetch)
# origin  https://github.com/rgaleanog/todo-list-aws.git (push)
```

```
git remote set-url origin https://github.com/user/todo-list-aws.git
# Cambiar las url remotas de 'origin'

git remote -v
# Verificar la nueva URL
# origin  https://github.com/user/todo-list-aws.git (fetch)
# origin  https://github.com/user/todo-list-aws.git (push)
```

Esta etapa comprenderá la labor de alojar el código final generado del caso práctico (que partía al inicio de una versión «base») en el servicio de control del histórico de versiones del software Github.

Observaremos acto seguido como se añade un nuevo proyecto sobre el explorador de carpetas (*toggletree*) situado en el lateral izquierdo de nuestra instancia del IDE Cloud9:



Los pasos serían los siguientes en el mismo orden en el que se sugieren ahora, a realizar del mismo modo sobre el terminal (bash) integrado en Cloud9:

- ▶ Añadir al commit los ficheros que apliquen al cambio a *commitear*:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git add <nombre  
fichero>  
vocstartsoft:~/environment/todo-list-aws (develop) $ git add <nombre  
fichero>  
...  
...
```

- ▶ Comentar el *commit* con un título representativo y añadir una descripción que detalle bien el cambio. Que sea lo extensa que haga falta siempre y cuando se entienda:
- ▶ Añadir al *commit* los ficheros que apliquen al cambio a *commitear*:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git commit
```

(Solo indicar git commit, esto permite editar el commit desde un editor y añadir la descripción).

- ▶ Hacer *push* del cambio:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git push origin  
develop
```

- ▶ Editar el fichero CHANGELOG.md con todos los cambios realizados en el código en anteriores pasos (en este caso, “Versión inicial de código”).

```
# Changelog  
All notable changes to this project will be documented in this file.  
  
The format is based on [Keep a
```

```
Changelog](https://keepachangelog.com/en/1.0.0/),  
and this project adheres to [Semantic  
Versioning](https://semver.org/spec/v2.0.0.html).
```

```
## [1.0.0] - 2020-08-05  
### Added  
- Versión inicial de código.
```

- ▶ Hacer *push* en rama DEVELOP:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git add  
CHANGELOG.md  
vocstartsoft:~/environment/todo-list-aws (develop) $ git commit  
...  
vocstartsoft:~/environment/todo-list-aws (develop) $ git push origin  
develop
```

- ▶ Hacer un *git merge* con la **rama MASTER**:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git checkout  
master  
vocstartsoft:~/environment/todo-list-aws (master) $ git merge develop  
...  
vocstartsoft:~/environment/todo-list-aws (develop) $ git push origin  
master
```

- ▶ Añadir un tag sobre el commit generado a partir del merge, en este caso siendo la primera versión del código:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git tag -a 1.0.0  
-m 'version 1.0.0'
```

- ▶ Hacer *push* del tag para que no se quede en local:

```
vocstartsoft:~/environment/todo-list-aws (develop) $ git push origin  
--tags
```

Tras llevar a cabo la migración del repositorio a la cuenta del alumno, realizar la siguiente comprobación de la URL de nueva creación de la ubicación destino:

Comando	Resultado
\$ git remote -v	Completar en documento final a entregar: Plantilla Solución CP1.docx

Validación de interfaz de SAM CLI (Command Line Interface) y análisis de repositorio

Esta sección fue abordada en buena parte al comienzo del enunciado, concretamente en *Setup inicial cuenta AWS*, y en *Creación de entorno de desarrollo con AWS Cloud9*.

Una de las herramientas en las que se apoya este apartado del caso práctico 1 es la interfaz de línea de comandos de **AWS SAM**.

Validación SAM

Se recomienda por ello validar la versión instalada de SAM CLI en el entorno de desarrollo Cloud9 o conectándose a la instancia desde el propio laboratorio:

Comando	Resultado
\$ sam --version	Completar en documento final a entregar: Plantilla Solución CP1.docx

Una vez asentadas las bases del proyecto, es tiempo de desarrollo en profundidad por parte del alumno.

La labor se concentra en:

- ▶ El nuevo proyecto, con nomenclatura ***todo-list-aws***, será el proyecto final para entregar por parte del alumno, y recogerá la consecución de los distintos hitos planteados en este apartado A. Este nuevo proyecto SAM deberá contemplar para darse por completada esta primera sección los siguientes requisitos en la especificación de la plantilla SAM ([template.yaml](#)):
 - Inclusión de definición de las 5 funciones Lambda correspondientes a cada uno de los métodos de que abordar la funcionalidad requerida para la aplicación de tareas (creación, eliminación, consulta, actualización y listado).
 - Definición de servicio, roles y parametrización de recursos (tablas requeridas) para aprovisionamiento de la información relativa a las tareas ToDo en AWS DynamoDB (más información [aquí](#)).

La estructura del nuevo proyecto deberá contener el esqueleto de proyecto SAM, conteniendo todos los ficheros fuente con la lógica a invocar por parte de las funciones Lambda definidas en la nueva plantilla SAM.

Análisis del repositorio

Se han de identificar los diferentes ficheros y directorios que contiene el repositorio y **explicar el sentido de cada uno de ellos**:

Ficheros	Contenido
src	Completar en documento final a entregar: Plantilla Solución CP1.docx
test	Completar en documento final a entregar: Plantilla Solución CP1.docx
Pipelines	Completar en documento final a entregar: Plantilla Solución CP1.docx
template.yaml	Completar en documento final a entregar: Plantilla Solución CP1.docx
samconfig.toml	Completar en documento final a entregar: Plantilla Solución CP1.docx
localEnvironment.json	Completar en documento final a entregar: Plantilla Solución CP1.docx

Observaciones: Se valorará positivamente labores de documentación: comentarios explicativos en el propio código, actualización de ficheros README.md y CHANGELOG.md con la explicación de las novedades presentadas en el evolutivo SW.

Ejecución de proyecto en entorno local (SAM CLI)

La labor del alumno en este apartado será la de ejecutar localmente el proyecto SAM de la lista de tareas implementado en la anterior sección en proyecto local ***todo-list-aws***, capturando el momento de ejecución y disponibilidad del servidor local, así como la invocación de distintas peticiones de servicio a cada uno de los métodos (6) de la solución RESTful, acreditando la realización de su completa funcionalidad. **Es conveniente precisar, llegados a este punto del caso práctico, que el esfuerzo y dedicación resultantes para su completitud forman parte del desempeño de operaciones diarias de la figura de DevOps. Es por ello por lo que el objetivo de la práctica es la de adquirir destreza en el contexto de trabajo y distinto uso de herramientas aquí descritas.**

Observaciones: De cara a la ejecución local de nuestra aplicación SAM, será conveniente la configuración de un entorno efímero de validación. Es por ello por lo que se hará uso de la funcionalidad **sam local** y la resolución de dependencias de persistencia de tareas en la todo-list por parte de la lógica de Backend (Servicio de DynamoDB sobre contenedores Docker). Además, hay que recordar que todas estas pruebas deberían de llevarse a cabo dentro de una nueva rama del repositorio, respetando el modelo de *branching* descrito previamente. Es decir, se ha de crear una nueva rama *feature* y trabajar desde ella para todas las pruebas que se trabajen con sam local. En este caso, al ejecutarse todo de manera local, será necesario

Más información: <https://hub.docker.com/r/amazon/dynamodb-local>

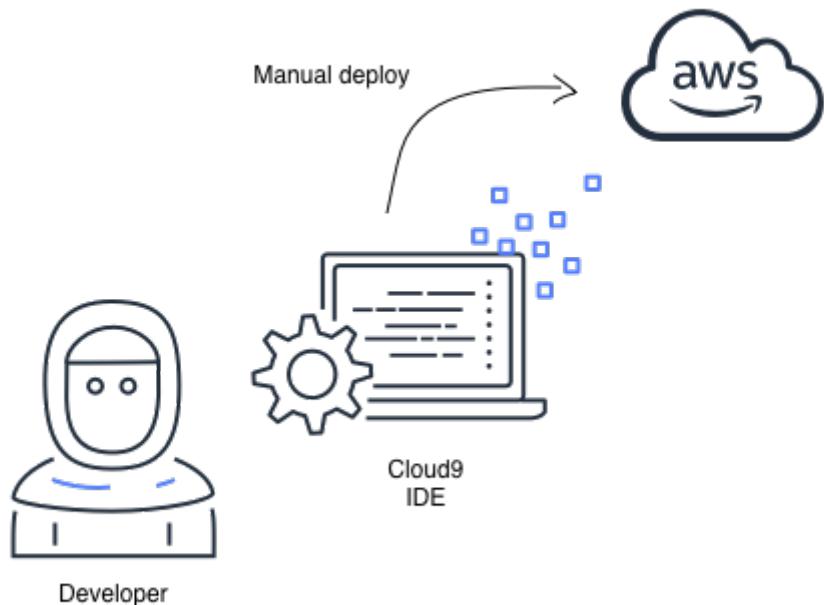
Para facilitar la misión de validación de este apartado, se adjunta el listado de peticiones a los métodos del servicio desplegados localmente con la url o endpoint relativa, y un campo «resultado» con objeto de ser completado por parte del alumno, así como los distintos pasos para conseguir desplegar sam local, como las evidencias de que se ha realizado correctamente (**para completar esta parte del caso práctico 1, es muy recomendable leer el fichero README.md disponible en el repositorio de la práctica**):

Pasos a realizar	Salida
Crear red de docker	Completar en documento final a entregar: Plantilla Solución CP1.docx
Levantar contenedor de docker	Completar en documento final a entregar: Plantilla Solución CP1.docx
Crear tabla en dynamodb local	Completar en documento final a entregar: Plantilla Solución CP1.docx
Empaquetar proyecto con SAM	Completar en documento final a entregar: Plantilla Solución CP1.docx
Levantar la API localmente	Completar en documento final a entregar: Plantilla Solución CP1.docx

Función	In/Out	Script
Create	comando	curl -X POST http://127.0.0.1:8081/todos --data '{ "text": "Learn Serverless" }'
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
List	comando	curl http://127.0.0.1:8081/todos
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
Get	comando	curl http://127.0.0.1:8081/todos/<id>
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
Update	comando	curl -X PUT http://127.0.0.1:8081/todos/<id> --data '{ "text": "Learn Serverless", "checked": true }'
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
Delete	comando	curl -X DELETE http://127.0.0.1:8081/todos/<id>
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx

Despliegue manual de aplicación SAM en Amazon Web Services

Tiempo para crear, empaquetar e implementar manualmente una aplicación Serverless mediante la CLI de AWS SAM. Esta tarea posibilita conocer los fundamentos necesarios, como etapa previa a la construcción de un *pipeline* de integración e integra continua totalmente automatizada.



- ▶ Artefactos: definición.

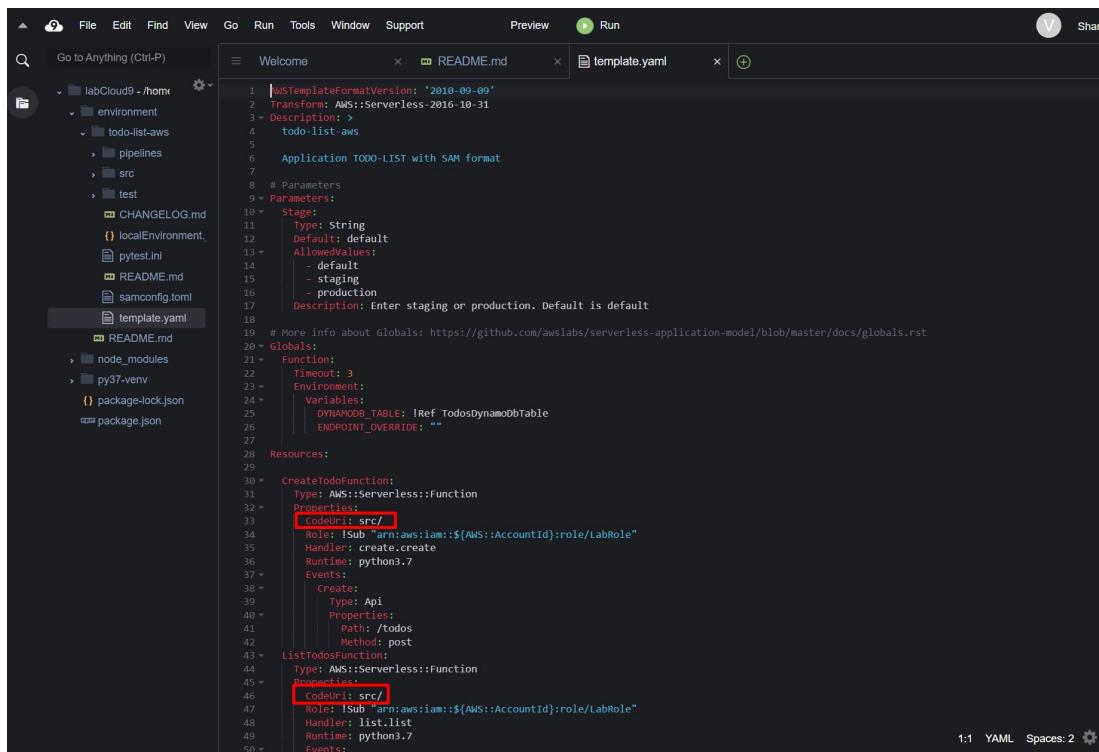
Los *artefactos* se refieren a la salida de un proceso de construcción en el contexto de CI / CD. Los artefactos suelen tener extensión .zip/.tar, .jar o (.bin) binario. Posteriormente, estos artefactos pueden implementarse (“instalarse”) en diferentes entornos industriales (es decir, DEV, PRE, PRO, etc.). En el supuesto de proyectos Serverless, dichos artefactos han de ser alojados en un bucket de S3 para el que el servicio Lambda pueda recogerlos. SAM CLI se encarga de administrar este proceso de carga de artefactos en S3 y hacer referencia a ellos en el momento de la implementación.

Se distinguen 2 artefactos en un proyecto Serverless: el primero de ellos contiene su código base, comprimido en formato .zip y depositándose en S3 automáticamente por AWS SAM CLI. El segundo artefacto que se genera en la fase de empaquetamiento también (al igual que el caso del primer artefacto) es la plantilla empaquetada, en este caso una copia de **template.yaml**, incluyendo una

referencia a la ubicación del archivo zip en un bucket de S3. La siguiente imagen muestra un ejemplo de una plantilla empaquetada, reflejado en el parámetro **CodeUri** con esa redirección en lugar de en un directorio local. Así es como AWS Lambda puede extraer su código en el momento (fase) de implementación.

► Construcción de la aplicación (**Build Stage**).

Para construir un proyecto SAM a través de AWS SAM CLI se hace uso del comando **sam build**. Este comando itera en busca de las funciones de su aplicación, buscando el archivo de manifiesto (tales como *requirements.txt* o *package.json*) que contienen las dependencias asociadas, creando automáticamente artefactos de implementación.



```
1 |WSTemplateFormatVersion: '2010-09-09'
2 Transform: AWS::Serverless-2016-10-31
3 -+ Description: >
4   todo-list-aws
5
6   Application TODO-LIST with SAM format
7
8 # Parameters
9 + Parameters:
10 + Stage:
11   Type: String
12   Default: default
13   AllowedValues:
14     - default
15     - staging
16     - production
17   Description: Enter staging or production. Default is default
18
19 # More info about Globals: https://github.com/awslabs/serverless-application-model/blob/master/docs/globals.rst
20 + Globals:
21   Function:
22     Timeout: 3
23   Environment:
24   Variables:
25     DYNAMODB_TABLE: !Ref TodosDynamoDBTable
26   ENDPOINT_OVERRIDE: ''
27
28 Resources:
29
30 + CreateTodoFunction:
31   Type: AWS::Serverless::Function
32   Properties:
33     Codeuri: src/
34     Role: !Sub "arn:aws:iam:${AWS::AccountId}:role/LabRole"
35     Handler: create.create
36     Runtime: python3.7
37   Events:
38     Create:
39       Type: Api
40       Properties:
41         Path: /todos
42         Method: post
43   ListTodosFunction:
44   Type: AWS::Serverless::Function
45   Properties:
46     Codeuri: src/
47     Role: !Sub "arn:aws:iam:${AWS::AccountId}:role/LabRole"
48     Handler: list.list
49     Runtime: python3.7
50   Events:
```

Continuando con la aplicación todo-list-aws nos situamos en ese nivel de carpeta de proyecto y ejecutamos el comando en cuestión:

```
voclabs:~/environment/todo-list-aws (feature) $ sam build
    SAM CLI now collects telemetry to better understand customer
needs.

    You can OPT OUT and disable telemetry collection by setting
the
    environment variable SAM_CLI_TELEMETRY=0 in your shell.
    Thanks for your help!

    Learn More: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-telemetry.html

Building codeuri: /home/ubuntu/environment/todo-list-aws/src runtime:
python3.7 metadata: {} architecture: x86_64 functions:
['CreateTodoFunction', 'ListTodosFunction', 'GetTodoFunction',
'UpdateTodoFunction', 'DeleteTodoFunction']
Running PythonPipBuilder:ResolveDependencies
Running PythonPipBuilder:CopySource

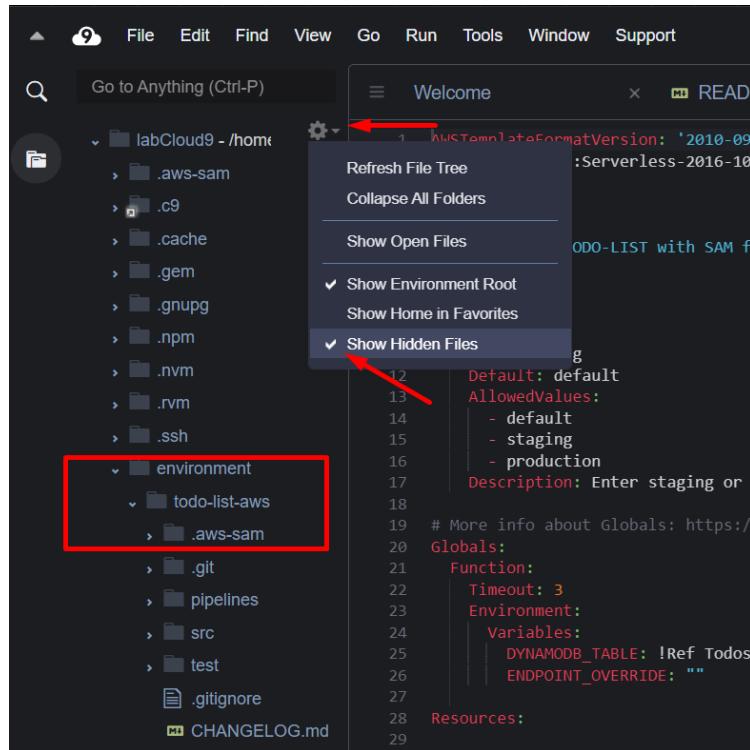
Build Succeeded

Built Artifacts  : .aws-sam/build
Built Template   : .aws-sam/build/template.yaml

Commands you can use next
=====
[*] Invoke Function: sam local invoke
[*] Deploy: sam deploy --guided

SAM CLI update available (1.36.0); (1.33.0 installed)
To download: https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-install.html
```

Cuando la compilación finalice correctamente, deberá de ver un nuevo directorio creado en la raíz del proyecto llamado **.aws-sam**. Es una carpeta oculta, por lo que, si desea verla en el IDE, asegúrese de habilitar *Mostrar archivos ocultos* en Cloud9 para verla, de la siguiente manera:



En la nueva carpeta generada en tiempo de compilación puede encontrarse el contenido requerido para el despliegue del aplicativo en cualquier entorno productivo: código fuente, dependencias definidas en fichero de requerimientos, así como la nueva plantilla de definición de la arquitectura software.

► Despliegue de la aplicación (**Deploy Stage**).

Una vez abordada la fase de compilación o construcción de los artefactos de nuestro proyecto Serverless, es tiempo de desplegar el mismo a través de la pila (stack) del servicio AWS CloudFormation. A través del comando **sam deploy** y su modo interactivo (especificando el parámetro `--guided` seguidamente) aprenderemos a conocer el detalle para nuestra tarea obligatoria y futuros despliegues de aplicativos.

Ejecutamos por tanto el comando en cuestión en el mismo nivel de directorio donde se encuentra `template.yaml` (el de origen durante el scaffolding de proyecto, no el generado durante el Build Stage):

```
voclabs:~/environment/todo-list-aws (feature) $ sam deploy --guided
```

Sería necesario, por tanto, debido al *flag* incluido en la ejecución del comando para habilitar el modo interactivo, introducir valores a determinados parámetros de entrada según necesidades del proyecto. Por defecto, se indican una serie de valores genéricos sugeridos:

```
voclabs:~/environment/todo-list-aws (feature) $ sam deploy --guided
Configuring SAM deploy
=====
Looking for config file [samconfig.toml] : Found
Reading default arguments : Success

Setting default arguments for 'sam deploy'
=====
Stack Name [todo-list-aws]: todo-list-aws
AWS Region [us-east-1]: us-east-1
Parameter Stage [default]: default
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
Confirm changes before deploy [y/n]: y
#SAM needs permission to be able to create roles to connect to the resources in your template
Allow SAM CLI IAM role creation [y/n]: y
CreateTodoFunction may not have authorization defined, Is this okay? [y/N]: y
ListTodosFunction may not have authorization defined, Is this okay? [y/N]: y
GetTodoFunction may not have authorization defined, Is this okay? [y/N]: y
UpdateTodoFunction may not have authorization defined, Is this okay? [y/N]: y
DeleteTodoFunction may not have authorization defined, Is this okay? [y/N]: y
Save arguments to configuration file [Y/n]: Y
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]:

Looking for resources needed for deployment:
Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-upt09go84tay
A different default S3 bucket can be set in samconfig.toml

Saved arguments to config file
Running 'sam deploy' for future deployments will use the parameters saved above.
Deploying with following valuesnged by modifying samconfig.toml
=====
syntax at
Stack name          : todo-list-aws:stack-model/latest/developerguide/serverless-sam-cli-config.html
Region             : us-east-1
Confirm changeset   : True
Deployment s3 bucket: aws-sam-cli-managed-default-samclisourcebucket-upt09go84tay
Capabilities       : ["CAPABILITY_IAM"]
Parameter overrides: {"Stage": "default"}
Signing Profiles    : {}
```

En algún momento, SAM solicitará la confirmación de la implementación. Esto es posible porque primero crea un CloudFormation ChangeSet y luego solicita confirmación para ejecutarlo. Esto es lo que se conoce como *dry run deployment*, y constituye una buena práctica cuando se realizan despliegues a través de CloudFormation.

Tras una serie de minutos, tiempo requerido en la creación de recursos sobre el proyecto SAM. El resultado que confirma la completa finalización del proceso debe arrojar un mensaje similar al siguiente:

```

CloudFormation stack changeset
-----
Operation          LogicalResourceId      ResourceType      Replacement
+ Add              CreateTodoFunctionCreatePermissionProd   AWS::Lambda::Permission    N/A
+ Add              CreateTodoFunction          AWS::Lambda::Function     N/A
+ Add              DeleteTodoFunctionCreatePermissionProd   AWS::Lambda::Permission    N/A
+ Add              DeleteTodoFunction          AWS::Lambda::Function     N/A
+ Add              GetTodoFunctionCreatePermissionProd   AWS::Lambda::Permission    N/A
+ Add              GetTodoFunction          AWS::Lambda::Function     N/A
+ Add              ListTodosFunctionCreatePermissionProd   AWS::Lambda::Permission    N/A
+ Add              ListTodosFunction          AWS::Lambda::Function     N/A
+ Add              Prod                  AWS::Lambda::Function     N/A
+ Add              ServerlessRestApiDeployment141b84        AWS::ApiGateway::Deployment N/A
+ Add              2de6                AWS::Apigateway::Stage      N/A
+ Add              ServerlessRestApi          AWS::Apigateway::RestApi    N/A
+ Add              TodosDynamoDbTable        AWS::DynamoDb::Table       N/A
+ Add              UpdateTodoFunctionCreatePermissionProd   AWS::Lambda::Permission    N/A
+ Add              UpdateTodoFunction          AWS::Lambda::Function     N/A
-----
Changeset created successfully. arn:aws:cloudformation:us-east-1:025379707399:changeSet:samcli-deploy1639222703/ed7abe49-b19d-4235-a890-e9309da6277
6

Previewing CloudFormation changeset before deployment
=====
Deploy this changeset? [y/N]: █

CREATE_IN_PROGRESS      AWS::Lambda::Function      CreateTodoFunction           Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Function      GetTodoFunction             Resource creation Initiated
CREATE_COMPLETE         AWS::Lambda::Function      CreateTodoFunction           -
CREATE_COMPLETE         AWS::Lambda::Function      UpdateTodoFunction          -
CREATE_COMPLETE         AWS::Lambda::Function      DeleteTodoFunction          -
CREATE_COMPLETE         AWS::Lambda::Function      GetTodoFunction             -
CREATE_COMPLETE         AWS::Lambda::Function      ListTodosFunction          -
CREATE_IN_PROGRESS      AWS::Apigateway::RestApi    ServerlessRestApi           -
CREATE_IN_PROGRESS      AWS::Apigateway::RestApi    ServerlessRestApi           Resource creation Initiated
CREATE_COMPLETE         AWS::Apigateway::RestApi    ServerlessRestApi           -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     CreateTodoFunctionCreatePermissionProd   -
CREATE_IN_PROGRESS      AWS::Apigateway::Deployment  ServerlessRestApiDeployment141b84  -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     2de6                         -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     GetTodoFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     CreateTodoFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     GetTodoFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     UpdateTodoFunctionCreatePermissionProd   -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     nProd                         -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     GetTodoFunctionCreatePermissionProd   -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     DeleteTodoFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     nProd                         -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     ListTodosFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     Prod                          -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     DeleteTodoFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     nProd                         -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     ListTodosFunctionCreatePermissionProd   Resource creation Initiated
CREATE_IN_PROGRESS      AWS::Lambda::Permission     Prod                          -
CREATE_IN_PROGRESS      AWS::Lambda::Permission     UpdateTodoFunctionCreatePermissionProd   Resource creation Initiated

```

CREATE_IN_PROGRESS	AWS::Lambda::Permission	UpdateTodoFunctionCreatePermissio	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment141b84	Resource creation Initiated
CREATE_COMPLETE	AWS::ApiGateway::Deployment	2de6	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	-
CREATE_COMPLETE	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	-
CREATE_IN_PROGRESS	AWS::Lambda::Permission	ServerlessRestApiDeployment141b84	Resource creation Initiated
CREATE_COMPLETE	AWS::Lambda::Permission	CreateTodoFunctionCreatePermissio	-
CREATE_COMPLETE	AWS::Lambda::Permission	nProd	-
CREATE_COMPLETE	AWS::Lambda::Permission	UpdateTodoFunctionCreatePermissio	-
CREATE_COMPLETE	AWS::Lambda::Permission	nProd	-
CREATE_COMPLETE	AWS::Lambda::Permission	GetTodoFunctionCreatePermissionPr	-
CREATE_COMPLETE	AWS::Lambda::Permission	od	-
CREATE_COMPLETE	AWS::Lambda::Permission	ListTodosFunctionCreatePermission	-
CREATE_COMPLETE	AWS::Lambda::Permission	Prod	-
CREATE_COMPLETE	AWS::CloudFormation::Stack	DeleteTodoFunctionCreatePermissio	-
CREATE_COMPLETE	AWS::CloudFormation::Stack	nProd	-
		todo-list-aws	-

CloudFormation outputs from deployed stack	
Outputs	
Key	BaseUrlApi
Description	Base URL of API
Value	https://fzaa8n1c3.execute-api.us-east-1.amazonaws.com/Prod
Key	DeleteTodoApi
Description	API Gateway endpoint URL for \${opt:stage} stage for Delete TODO
Value	https://fzaa8n1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/{id}
Key	ListTodosApi
Description	API Gateway endpoint URL for \${opt:stage} stage for List TODO
Value	https://fzaa8n1c3.execute-api.us-east-1.amazonaws.com/Prod/todos
Key	UpdateTodoApi
Description	API Gateway endpoint URL for \${opt:stage} stage for Update TODO
Value	https://fzaa8n1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/{id}
Key	GetTodoApi
Description	API Gateway endpoint URL for \${opt:stage} stage for Get TODO
Value	https://fzaa8n1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/{id}
Key	CreateTodoApi
Description	API Gateway endpoint URL for \${opt:stage} stage for Create TODO
Value	https://fzaa8n1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/

Successfully created/updated stack - todo-list-aws in us-east-1

¿Qué sucedió internamente en el proceso de despliegue? De cara a ayudar a facilitar la comprensión del alumno de todo proceso de despliegue manual de una arquitectura de servicios que constituye una aplicación en un entorno Cloud como AWS, destacamos las siguientes realizadas:

- ▶ El código fuente (software base) se empaqueta como artefacto en formato de archivo .zip.
- ▶ SAM crea un bucket específico en S3 en su cuenta AWS Academy (en caso de no existir). **NOTA:** El repositorio tiene incluido un bucket por defecto a modo de ejemplo, el alumno debe de crear un repositorio específico o dejar que sam cree uno por él, pero necesita revisar esto cuando lo ejecute.
- ▶ El artefacto en formato .zip se aloja en el bucket de S3 anteriormente creado.
- ▶ SAM crea la plantilla «empaquetada» (una evolución del fichero template.yaml) que hace referencia a la ubicación del archivo zip en S3 en vez de local.
- ▶ Esta plantilla también se deposita en el mismo bucket de S3.

- SAM inicia el despliegue de la aplicación a través de CloudFormation ChangeSets.

Amazon S3 > aws-sam-cli-managed-default-samclisourcebucket-up09go84tay > todo-list-aws/

Objetos (8)

Los objetos son las entidades fundamentales que se almacenan en Amazon S3. Puede utilizar el inventario de Amazon S3 para obtener una lista de todos los objetos de su bucket. Para que otras personas obtengan acceso a sus objetos, tendrá que concederles permisos de forma explícita. [Más información](#)

Nombre	Tipo	Última modificación	Tamaño	Clase de almacenamiento
a80d5a359f57939b697d06d62989eb44.template	template	11 Dec 2021 1:03:13 AM CET	4.2 KB	Estándar
85ed6beab8beef39ec1b5828d641fe0	-	11 Dec 2021 1:03:12 AM CET	443.7 KB	Estándar

Observación: La primera vez que se lleva a cabo un despliegue guiado, se crea un nuevo archivo en la raíz del proyecto SAM con los parámetros de despliegue especificados. En próximos despliegues ante cambios en el proyecto SAM (utilizando nuevamente el comando **sam deploy**), recurrirá a este fichero sin tener que ingresarlos nuevamente.

```

1 version = 0.1
2 [default]
3 [default.deploy]
4 [default.deploy.parameters]
5 stack_name = "todo-list-aws"
6 s3_bucket = "aws-sam-cli-managed-default-samclisourcebucket-up09go84tay"
7 s3_prefix = "todo-list-aws"
8 region = "us-east-1"
9 capabilities = "CAPABILITY_IAM"
10 parameter_overrides = "Stage=\"default\""
11 image_repositories = []
12 confirm_changeset = true

```

Para más información sobre despliegues guiados, he aquí [más información](#).

Recomendado: Una vez desplegada la aplicación en nuestra cuenta de alumno AWS Academy, es aconsejable verificar que está en proceso de ejecución, operativa dando el servicio para el que se propuso. Para ello debemos dirigirnos a la consola del servicio de **CloudFormation** (disponible al comienzo del caso práctico en la herramienta de PIN, o a través de la barra de búsqueda existente en la pestaña de **Services**). En el listado de stacks listados, debería de encontrarse uno de nombre **todo-list-aws** en estado *CREATE_COMPLETE*:

Nombre de la pila	Estado	Hora de creación	Descripción
todo-list-aws	CREATE_COMPLETE	2021-12-11 12:33:07 UTC+0100	todo-list-aws Application TODO-LIST with SAM format
todo-list-aws-production	CREATE_COMPLETE	2021-12-11 01:22:57 UTC+0100	todo-list-aws Application TODO-LIST with SAM format
todo-list-aws-staging	CREATE_COMPLETE	2021-12-11 01:20:28 UTC+0100	todo-list-aws Application TODO-LIST with SAM format

Tras comprobar la existencia del nuevo stack desplegado en el Cloud de AWS con el ejemplo de prueba de proyecto SAM, validaremos su correcto funcionamiento. Para ello, haremos clic en el stack **todo-list-aws** y luego sobre la pestaña *Outputs* o Salidas. En esta pestaña puede recogerse el detalle de salidas definidas en la plantilla SAM (template.yaml): URL de API Gateway, ARN de la función Lambda y del rol de IAM para la función.

Clave	Valor	Descripción	Nombre de exportación
BaseUrlApi	https://fzaa8ni1c3.execute-api.us-east-1.amazonaws.com/Prod	Base URL of API	-
CreateTodoApi	https://fzaa8ni1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/	API Gateway endpoint URL for \${opt:stage} stage for Create TODO	-
DeleteTodoApi	https://fzaa8ni1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/{id}	API Gateway endpoint URL for \${opt:stage} stage for Delete TODO	-
GetTodoApi	https://fzaa8ni1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/{id}	API Gateway endpoint URL for \${opt:stage} stage for Get TODO	-
ListTodosApi	https://fzaa8ni1c3.execute-api.us-east-1.amazonaws.com/Prod/todos	API Gateway endpoint URL for \${opt:stage} stage for List TODO	-
UpdateTodoApi	https://fzaa8ni1c3.execute-api.us-east-1.amazonaws.com/Prod/todos/{id}	API Gateway endpoint URL for \${opt:stage} stage for Update TODO	-

Una vez desplegada la API, el alumno deberá de completar la siguiente sección:

Observaciones: Se valorará positivamente la generación de un reporte detallado del proceso de obtención de los distintos hitos: capturas de pantalla acreditando la consecución de los objetivos, diagramas explicativos, etc.

Complete nuevamente el siguiente cuadro con los *endpoints* de cada función:

Función	Endpoint
Create	https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos
List	https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos
Get	<a href="https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>">https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>
Update	<a href="https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>">https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>
Delete	<a href="https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>">https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>

Una vez finalizado el despliegue manual, se ha de validar que los *endpoints* responden adecuadamente a las llamadas que se hagan a través de API Gateway, mediante invocaciones curl. Incluye en la siguiente tabla los comandos curl realizados y los resultados obtenidos. **En caso de error**, es conveniente redesplegar de nuevo, por si fuera un error temporal durante el despliegue, ejecutando de nuevo **sam deploy**.

Adjuntar a continuación los resultados de las validaciones, así como los logs de los distintos comandos ejecutados:

Función	In/Out	Script
Create	comando	curl -X POST https://XXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos --data '{ "text": "Learn Serverless" }'
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
List	comando	curl https://XXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
Get	comando	curl https://XXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
Update	comando	curl -X PUT https://XXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id> --data '{ "text": "Learn python and more", "checked": true }'
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx
Delete	comando	curl -X DELETE https://XXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>
	resultado	Completar en documento final a entregar: Plantilla Solución CP1.docx

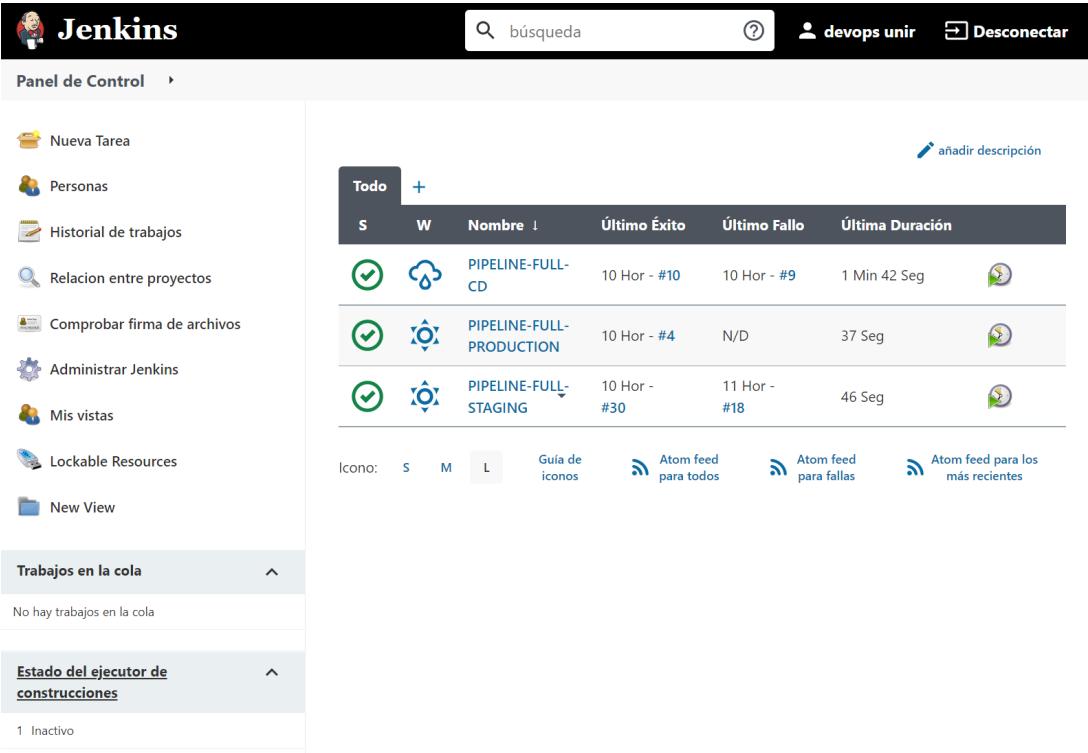
Resultados a mostrar	Salida
Log sam build	Completar en documento final a entregar: Plantilla Solución CP1.docx
Log sam deploy	Completar en documento final a entregar: Plantilla Solución CP1.docx

Creación de *pipelines* de Jenkins para despliegue de arquitectura completa

Una vez se ha procedido al aprovisionamiento y configuración de un servidor de integración continua Jenkins sobre una instancia EC2 con recursos computacionales

acorde a las necesidades del caso práctico, es tiempo de conocer qué recursos preconfigurados incluye la AMI de partida.

A continuación, se distinguen distintos tipos de *pipelines* de procesos, los cuales completan la finalidad de disponer de un mecanismo de automatización del despliegue de infraestructura multientorno, previa revisión de la lógica de funcionalidad ante evolutivos o correctivos del software que lo habilita.



The screenshot shows the Jenkins dashboard with the following details:

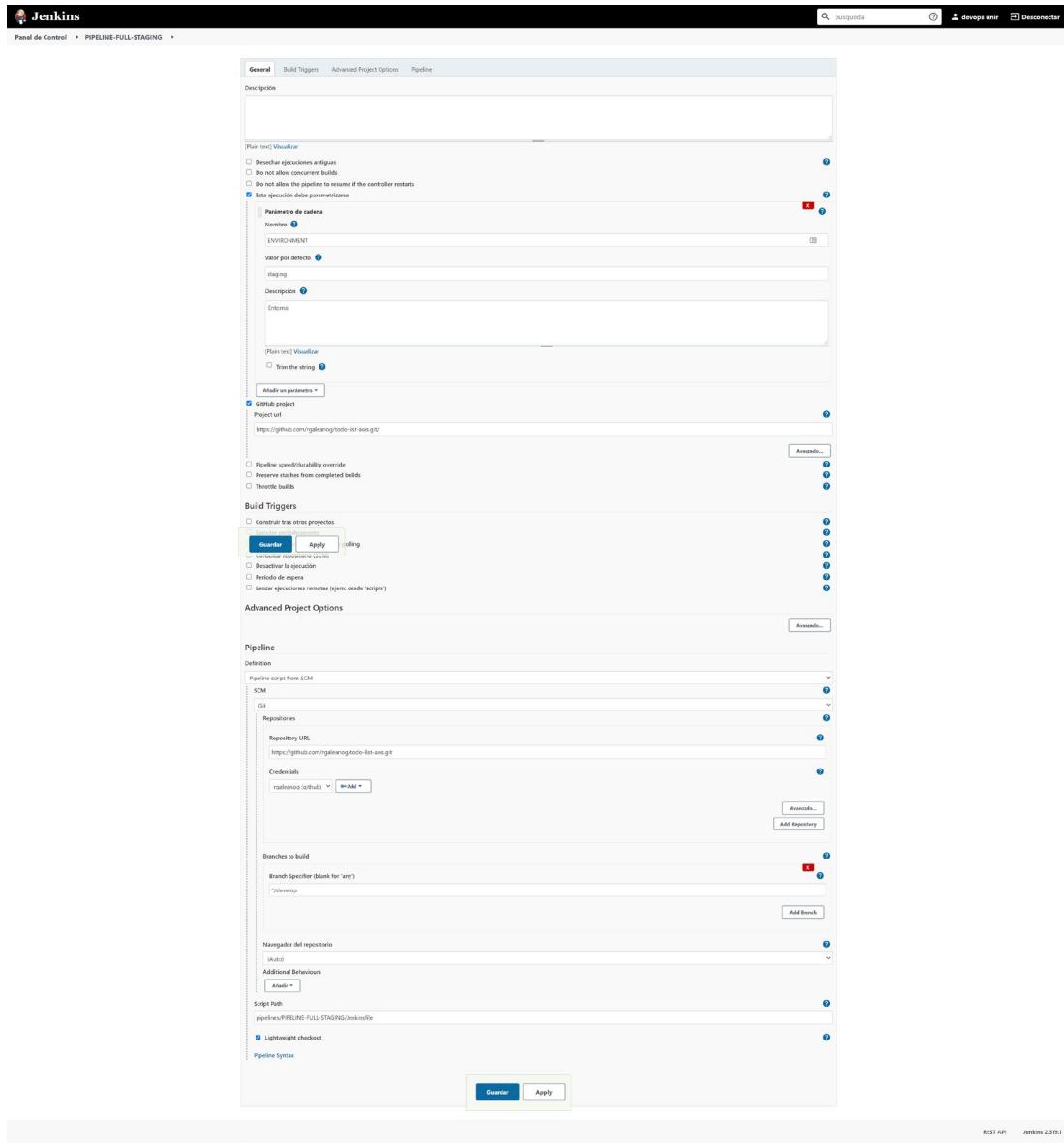
- Left Sidebar:** Includes links for "Nueva Tarea", "Personas", "Historial de trabajos", "Relacion entre proyectos", "Comprobar firma de archivos", "Administrador Jenkins", "Mis vistas", "Lockable Resources", and "New View".
- Top Bar:** Shows the Jenkins logo, user "devops unir", and a "Desconectar" button.
- Search Bar:** A search field with placeholder "búsqueda".
- Pipeline List:** A table listing three pipelines:

S	W	Nombre	Último Éxito	Último Fallo	Última Duración
✓	Cloud icon	PIPELINE-FULL-CD	10 Hor - #10	10 Hor - #9	1 Min 42 Seg
✓	Cloud icon	PIPELINE-FULL-PRODUCTION	10 Hor - #4	N/D	37 Seg
✓	Cloud icon	PIPELINE-FULL-STAGING	10 Hor - #30	11 Hor - #18	46 Seg
- Bottom Buttons:** Icons for "Icono: S M L", "Guía de iconos", "Atom feed para todos", "Atom feed para fallas", and "Atom feed para los más recientes".

Los tres *pipelines* que han de ser creados son los que tendrá que usar y configurar el alumno durante la práctica:

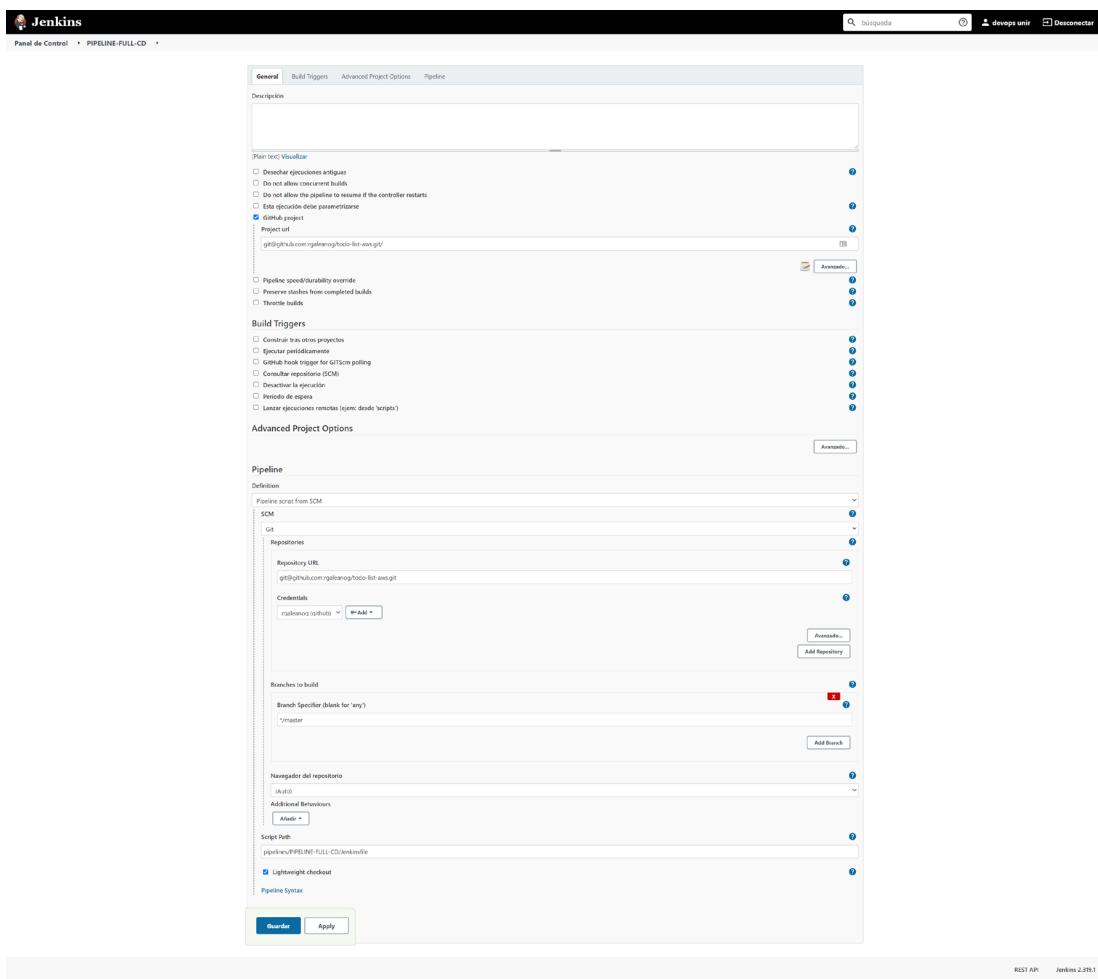
- ▶ **PIPELINE-FULL-STAGING:** *pipeline* sin configurar en Jenkins. Se proporciona el fichero Jenkinsfile en Groovy en el repositorio de la práctica. El pipeline debe de tomar un repositorio de Github con las credenciales del alumno configuradas para acceder por ssh a Github y cargar el pipeline habilitado. Además, este pipeline requiere de parametrización, con lo cual el alumno debe de crear una variable

denominada **ENVIRONMENT** para indicar el entorno que se está ejecutando. El valor de la variable será **staging**, para que el pipeline funcione correctamente.



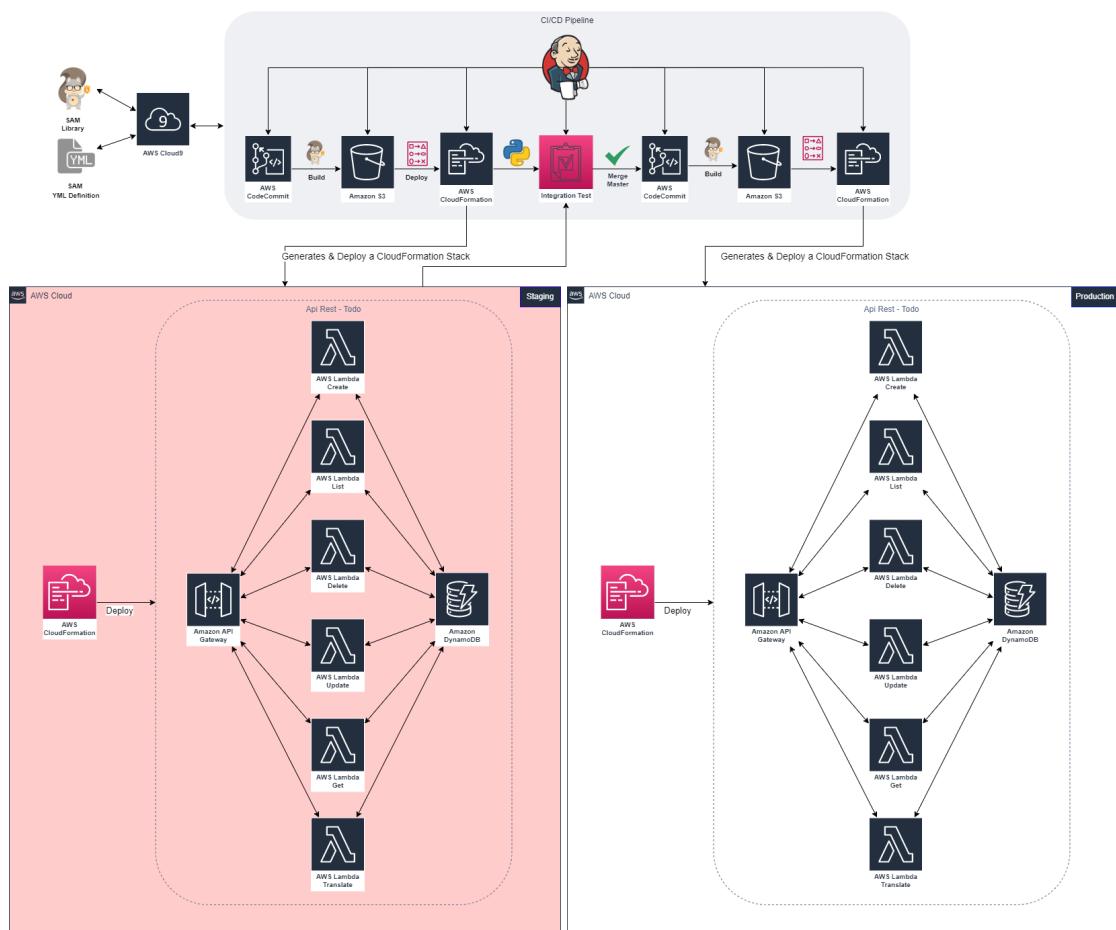
- ▶ **PIPELINE-FULL-PRODUCTION:** *pipeline* sin configurar en Jenkins. Se proporciona el fichero Jenkinsfile en Groovy en el repositorio de la práctica. El pipeline debe de tomar un repositorio de Github con las credenciales del alumno configuradas para acceder por ssh a Github y cargar el pipeline habilitado. Además, este pipeline requiere de parametrización, con lo cual el alumno debe de crear una variable denominada **ENVIRONMENT** para indicar el entorno que se está ejecutando. El valor de la variable será **production**, para que el pipeline funcione correctamente.

- ▶ **PIPELINE-FULL-CD:** este *pipeline* es el encargado de enganchar los *pipelines* de *staging* y *production*, con el objetivo de que el alumno sea capaz de completar un ciclo de despliegue continuo desde un commit al repositorio de manera automática.



Una vez configurados los Jobs de Jenkins, se han de construir los pipelines de los tres procesos. Como se indicaba anteriormente, los Jobs de los pipelines, denominados Jenkinsfile, ya se proporcionan en el repositorio de la práctica, por lo que la labor del alumno será la de validar que todo funciona, contestar a las preguntas que se le formulen y elevar la calidad del código para pasar del 77% al 80% y así poder pasar las pruebas de calidad.

Pipeline de Staging



El primer *pipeline* que debe de construir el alumno usando el proyecto de SAM, es el de despliegue del entorno de *staging*. Este entorno de *staging* disponibiliza a partir del repositorio de Github la infraestructura de la API de **todo-list-aws**. Por tanto, el alumno debe de implementar un *pipeline* de Jenkins denominado **PIPELINE-FULL-STAGING** cuyas etapas son:

- 1. Etapa de checkout:** vinculación de repositorio al objeto de operativización, constituyendo la etapa de configuración de orígenes del software, el cual contiene el conjunto de instrucciones de funcionalidad e infraestructura de servicios en AWS necesarios para ello. Para esta primera etapa el usuario debe de hacer un *checkout* de git para descargarse el repositorio al *workspace* de Jenkins desde Github. La rama que se ha de probar en este entorno es la **rama de develop**.

- 2. Etapa de setup del entorno:** después de descargarse el código fuente al contexto de Jenkins, el alumno deberá de configurar el entorno de manera adecuada para adaptar el entorno de Python con las dependencias necesarias para ejecutar el resto de tareas del *pipeline*. Para esto el alumno puede contemplar dos alternativas: a) utilizar [entornos virtuales de Python](#) o b) usar contenedores Docker de Python para aislar las dependencias en las cuáles el alumno debería de tener presente las dependencias con aws-cli, sam-cli y Python 3.7. Se aconseja la primera opción por simplicidad y optimización de recursos.
- 3. Etapa de test:** una vez el repositorio es descargado al espacio de trabajo de Jenkins y configurado el entorno adaptado adecuadamente, el alumno deberá de realizar las siguientes tareas:

- Pruebas de revisión estática de código. Para ello deben usar la librería [radon](#) de Python. En este caso se debe asegurar que todos los módulos de Python desarrollados tengan al menos una calidad de B o superior (de una escala de A a F) en cuanto a complejidad ciclomática. Si la calidad está por debajo de C, el *pipeline* debe fallar indicando la razón del error.

Scope: todo el código desarrollado en Python.

- Pruebas de calidad del código. El alumno deberá de validar que el código desarrollado en Python cumple con las reglas de estilo definidas por [pep8](#). Para ello el alumno deberá hacer uso de librerías que analicen esta calidad del código como [flake8](#) y corregir todos aquellos fallos detectados que no cumplan con las guías de estilo de Python definidos por [pep8](#). En caso de haber fallos de estilo el *pipeline* debe fallar indicando la razón del error.

Scope: todo el código desarrollado en Python.

- Pruebas de seguridad en el código. El alumno deberá de validar que no existen fallas de seguridad en el código de Python desarrollado. Para ello se usará la

librería [bandit](#). En caso de haber fallos de nivel alto el *pipeline* debe fallar indicando qué líneas de código tienen potenciales riesgos de seguridad.

Scope: todo el código desarrollado en Python.

- Pruebas unitarias. El objetivo será hacer las pruebas unitarias sobre librería de acceso a la base de datos de DynamoDB, con los métodos utilizados en la práctica, como se había explicado anteriormente. Es decir, métodos de put, get, update y delete sobre entradas en la base de datos de DynamoDB. Las pruebas unitarias se ejecutarán sobre las funciones de esta clase escrita en Python y se deberán de usar librerías como [pytest](#) o [unittest](#) para las pruebas unitarias. Se recomienda el uso de la librería [moto](#), para mockear las llamadas a la API de DynamoDB para la realización de las pruebas unitarias. Además de pasar sin fallo las pruebas definidas también, hará falta finalmente ejecutar una prueba de cobertura del código, para analizar la calidad de estas pruebas unitarias sobre la clase de acceso a la tabla de DynamoDB. Para ello se puede utilizar la librería [coverage](#) de Python. Para pasar a la siguiente etapa del *pipeline*, todas las pruebas unitarias han de ser pasadas y el informe de cobertura de código global debe de superar el **80%** para poder continuar. Por debajo del **80%** deberá fallar el *pipeline*. El código facilitado actualmente, contempla está limitado al 70% para que no falle el pipeline cuando el alumno lo pruebe. **Será labor del alumno realizar los cambios necesarios en el código para subir el porcentaje que falte al 80%, modificando las pruebas para pasar ese 80% que se solicita en la práctica.**

Scope: exclusivamente la librería de acceso a la tabla de la base de datos.

4. **Etapa de build:** una vez pasadas todas las pruebas, se deberá de construir el artefacto a desplegar en AWS mediante el comando **sam build**. Para ello el alumno debe de analizar la [guía de referencia de SAM](#) para poder invocarlo dentro del *pipeline* de Jenkins. El alumno deberá de tener presente en el fichero de

samconfig.toml de apuntar al bucket de S3 específico que haya creado para el despliegue de la práctica.

5. **Etapa de deploy:** una vez empaquetada la release de la arquitectura con SAM, para el entorno de *staging*, se ha de proceder al *deployment* de esta arquitectura. Si todas las pruebas son correctas y el código tiene una calidad adecuada, se ejecuta el comando de sam deploy, tal y como se indica en la documentación de [SAM](#). Es posible que sea necesario modificar la plantilla de SAM para poder parametrizarla para el entorno de *staging* y *production*.
6. **Etapa de prueba de integración:** finalmente, después de desplegar la arquitectura en el entorno de *Staging*, el alumno deberá de desarrollar una serie de pruebas de integración de los servicios desplegados en AWS, para poder analizar de manera automática si los servicios se han desplegado correctamente. Estas pruebas se pueden llevar a cabo nuevamente también con la librería de unittest o pytest. El alumno deberá de analizar y decidir cómo implementar estas pruebas de acuerdo con que, si se siguen los pasos naturales del API, los resultados devueltos sean los esperados. Las pruebas de integración serían:
 - Prueba que haga una llamada a la función **create** y el resultado de la respuesta http sea un 200 y que al consultar directamente contra la base de datos sea correcto.
 - Prueba que haga una llamada a la función **list** y el resultado de la respuesta http sea un 200 y que al consultar que el número de ítems sea distinto de cero.
 - Prueba que haga una llamada a la función **get/{id}**, tomando como referencia el identificador **id** generado en el punto a) de este apartado y validando que sea el mismo resultado del payload que el payload que se usó originalmente en el punto a) y que la respuesta http sea un 200.
 - Prueba que haga una llamada a la función **update/{id}** y el resultado de la respuesta http un 200 y que el valor actualizado sea exactamente el mismo, invocando a la función **get/{id}** y comparando los valores.

- Prueba que haga una llamada a la función **delete/{id}**, tomando como referencia el identificador **id** generado en el punto a) de este apartado y devuelva una respuesta 200. Después se debe de hacer una llamada **get/{id}** nuevamente con el id original para validar que ya no existe esa entrada en la tabla de la base de datos.

7. Etapa de limpieza del entorno: el alumno deberá dejar el entorno de Jenkins limpio de cara a futuras ejecuciones y también para optimizar el uso del disco de la instancia de Jenkins. Por tanto, se ha de implementar [algún tipo de mecanismo](#) que lleve a cabo esta tarea, ya sea utilizando métodos que ya proporcione Jenkins o de manera manual.

Una vez finalizadas estas pruebas, el entorno de *staging* quedaría desplegado y con las pruebas de integración ejecutadas. En caso de fallo del *pipeline*, se deberá de analizar y corregir los fallos. Al estar en un entorno de *staging* no sería necesario hacer *rollback* de la arquitectura, con el fin de analizar los posibles fallos y corregirlos.

Resultados a mostrar	Salida
Log <i>pipeline</i>	Completar en documento final a entregar: Plantilla Solución CP1.docx
Captura de pantalla del <i>pipeline</i>	Completar en documento final a entregar: Plantilla Solución CP1.docx
Comentarios adicionales	Completar en documento final a entregar: Plantilla Solución CP1.docx

A continuación, se pasaría a construir el *pipeline* del entorno de *production*.

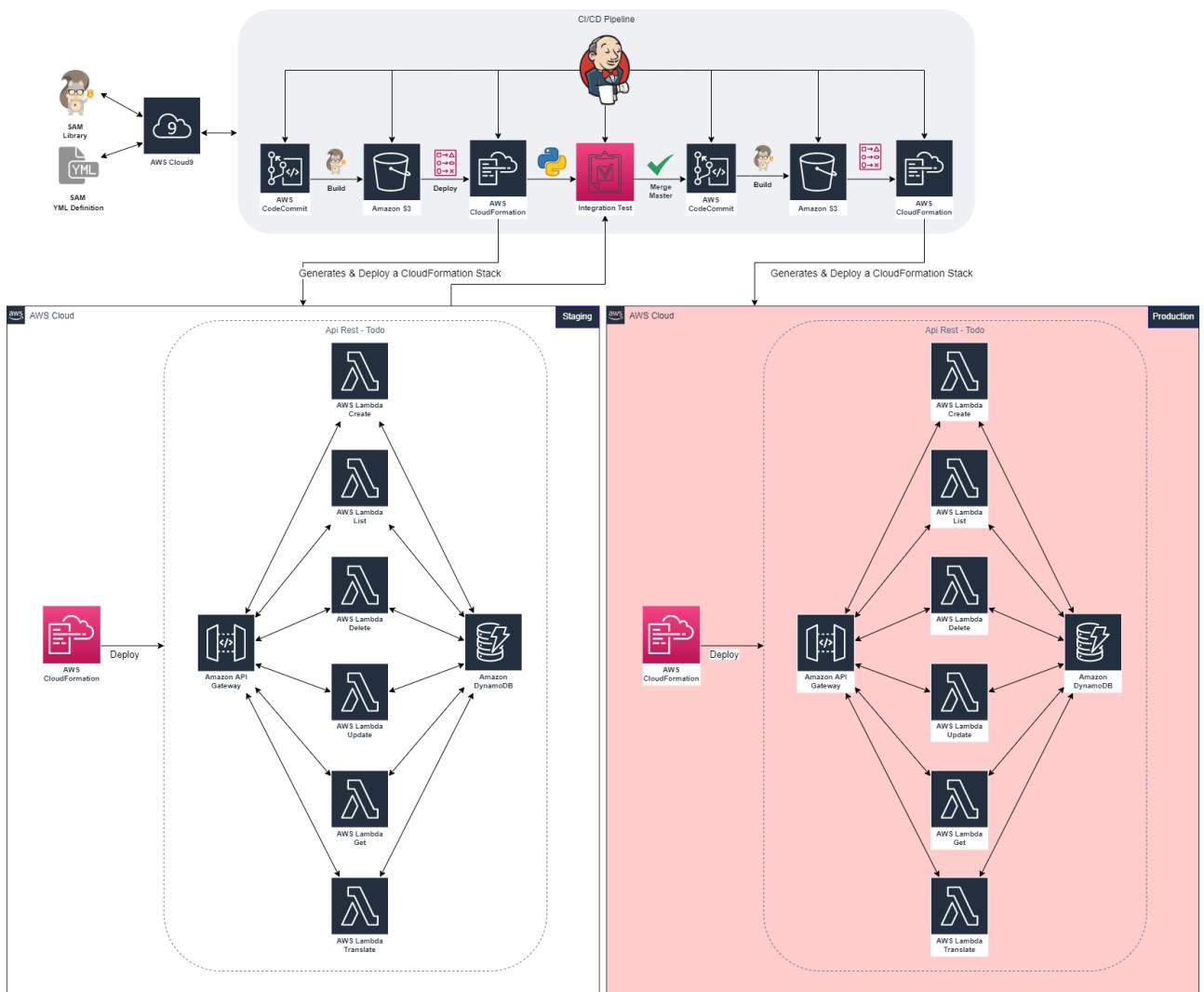
Pipeline de Production

Este entorno disponibiliza el entorno de producción dentro de AWS, usando SAM. Para ello lo ideal sería que este *pipeline* se construyese con un disparador que detecte cambios ante un merge de la rama de **develop a master**. Sin embargo, debido a la

naturaleza de la cuenta de AWS y a la poca duración de las credenciales de la cuenta de AWS Academy, este evento se generará de manera manual.

Por tanto, el alumno debe de implementar un *pipeline* de Jenkins denominado **PIPELINE-FULL-PRODUCTION** cuyas etapas son:

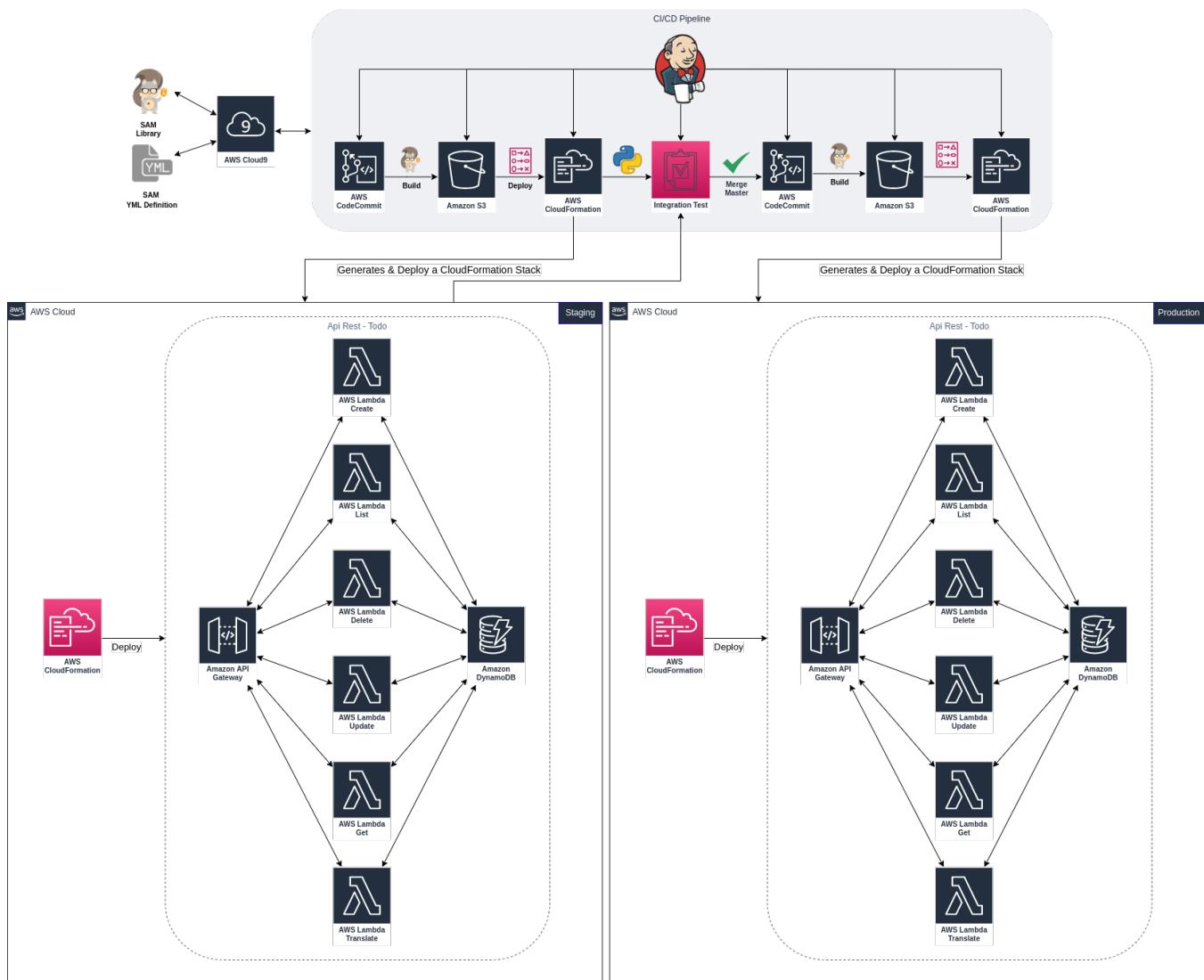
- 1. Etapa de checkout:** vinculación de repositorio al objeto de operativización, constituyendo la etapa de configuración de orígenes del software, el cual contiene el conjunto de instrucciones de funcionalidad e infraestructura de servicios en AWS necesarios para ello. Para esta primera etapa el usuario debe de hacer un *checkout* de git para descargarse el repositorio al *workspace* de Jenkins desde Github. La rama que se ha de probar en este entorno es la **rama de master**.



- 2. Etapa de setup del entorno:** después de descargarse el código fuente al contexto de Jenkins, el alumno deberá de configurar el entorno de manera adecuada para adaptar el entorno de Python con las dependencias necesarias para ejecutar el resto de las tareas del *pipeline*. Para esto el alumno puede contemplar dos alternativas, utilizar entornos virtuales de Python o usar contenedores Docker de Python para aislar las dependencias. Se aconseja la primera opción por simplicidad y optimización de recursos.
- 3. Etapa de build:** puesto que en el entorno de *staging* se pasaron todas las pruebas, se deberá de construir el artefacto a desplegar en AWS mediante el comando **sam package** para el entorno de *production*. El alumno deberá de tener presente en el fichero de **samconfig.toml** de apuntar al bucket de S3 específico que haya creado para el despliegue de la práctica.
- 4. Etapa de deploy:** una vez empaquetada la *release* de la arquitectura con SAM, para el entorno de *production*, se ha de proceder al *deployment* de esta arquitectura mediante el **sam deploy**. Si durante la ejecución de este despliegue falla el **sam deploy**, se producirá un *rollback* automático de la plantilla de CloudFormation, dejando la arquitectura como estaba antes del cambio.
- 5. Etapa de pruebas de integración:** como medida de control, se implementará de nuevo las pruebas de integración en el entorno de *production* de la misma manera que en el de *staging*. Además de implementar estas pruebas, el alumno debe de plantear una solución a cómo implementaría una solución de *rollback* en caso de que exista algún fallo que no se haya detectado en entornos previos, ni se haya detectado en tiempo de despliegue a través de la plantilla de CloudFormation. **No es necesario implementar el rollback en sí**, simplemente plantear una solución teórica según todo lo aprendido durante el desarrollo de la práctica.
- 6. Etapa de limpieza del entorno:** el alumno deberá dejar el entorno de Jenkins limpio de cara a futuras ejecuciones y también para optimizar el uso del disco de la instancia de Jenkins, de la misma manera que en *pipeline* del entorno de *staging*.

Resultados a mostrar	Salida
Log pipeline	Completar en documento final a entregar: Plantilla Solución CP1.docx
Captura de pantalla del pipeline	Completar en documento final a entregar: Plantilla Solución CP1.docx
Explicación teórica de rollback	Completar en documento final a entregar: Plantilla Solución CP1.docx
Comentarios adicionales	Completar en documento final a entregar: Plantilla Solución CP1.docx

CI/CD completo



En este punto el alumno simplemente debe integrar un *pipeline* que relacione ambos entornos de manera automática, produciendo un despliegue de la arquitectura desde el commit de un cambio. Para ello, se ha de definir un *pipeline* en Jenkins llamado **PIPELINE-FULL-CD** que relacione los *pipelines* de *staging* y *production*, y genere los cambios que considere necesarios en los repositorios de Github, como, por ejemplo, el *merge* automático de develop a master, si las pruebas de integración han sido exitosas y el disparo del evento de *pipeline* para la construcción del entorno de production a partir de la rama master del repositorio.

Resultados a mostrar	Salida
Log <i>pipeline</i>	Completar en documento final a entregar: Plantilla Solución CP1.docx
Captura de pantalla del <i>pipeline</i>	Completar en documento final a entregar: Plantilla Solución CP1.docx
Comentarios adicionales	Completar en documento final a entregar: Plantilla Solución CP1.docx

Caso práctico 1. Apartado B

En este apartado se han de desarrollar una serie de conclusiones la aproximación realizada durante el apartado A.

Además de las conclusiones, el alumno debe de analizar alternativas a esta solución despliegues realizados mediante SAM y mediante un entorno Jenkins. Para ello se propone que analice el uso completo del stack de AWS (CodeCommit, CodeBuild, CodeDeploy y *CodePipeline*) para identificar pros y contras y qué funcionalidades se podrían mejorar del ciclo de integración y despliegue continuos si se utilizaran y que planteen un *pipeline* teórico, identificando cada una de las etapas y qué servicios usarían para mejorar los que se han desarrollado durante la práctica. También sería interesante valorar alternativas Serverless como Serverless Framework, uso de herramientas como Sonarqube, etcétera.

Resultados a mostrar	Salida
Conclusiones finales	Completar en documento final a entregar: Plantilla Solución CP1.docx
Pros	Completar en documento final a entregar: Plantilla Solución CP1.docx
Contras	Completar en documento final a entregar: Plantilla Solución CP1.docx
Arquitectura alternativa explicación	Completar en documento final a entregar: Plantilla Solución CP1.docx
Arquitectura alternativa diagramas	Completar en documento final a entregar: Plantilla Solución CP1.docx

Caso práctico 1. Apartado C

Como se indicaba en el enunciado de la práctica, a continuación, se ha de desarrollar una nueva función lambda desde cero, partiendo de todo el conocimiento adquirido durante esta primera parte de la práctica. Recordad que esta función lambda debe devolver una entrada de la ToDo list, traducida al idioma que se solicite a través del API. Para ello se recomienda el uso de las API's de los servicios de [Comprehend](#) y [Translate de AWS](#) si fueran necesarios. Una vez desarrollada la nueva función lambda, se ha de integrar con el resto de componentes del API. Para ello habrá que incluir en el fichero **template.yml** la definición de la nueva lambda, el código fuente de esta función en el sitio adecuado de la estructura de directorios e integrarlo dentro del *pipeline* de CI/CD que se ha definido previamente, para ver cómo se propagan todos los cambios. El nuevo método de la API debe tener una estructura de este tipo:

- ▶ Método: GET
- ▶ PATH: /todos/<id>/<language>

A continuación, se ha de adjuntar la respuesta de tres invocaciones a la API, una en el idioma original del registro y dos con dos idiomas distintos (**Nota:** debe de funcionar con cualquier idioma que soporte la API de Translate de AWS y que contenga alfabeto latino):

Método	Resultado
/todos/<id>	Completar en documento final a entregar: Plantilla Solución CP1.docx
/todos/<id>/en	Completar en documento final a entregar: Plantilla Solución CP1.docx
/todos/<id>/fr	Completar en documento final a entregar: Plantilla Solución CP1.docx

Adicionalmente también se han de mostrar el código fuente desarrollado en la siguiente tabla, con el fin de tener la evidencia del trabajo realizado:

Fichero	Contenido
translate.py	Completar en documento final a entregar: Plantilla Solución CP1.docx

Pruebas

Se deberán de incluir dentro de las pruebas unitarias y de integración la nueva función translate, y validarla dentro del pipeline de CI/CD diseñado en el apartado A.

Para este último punto es de carácter opcional, se deja a discreción del alumno identificar las tareas que ha realizado de las pruebas, con qué tecnología ha conseguido implementar las pruebas dentro del CI/CD de SAM y qué problemas ha tenido para validar las API's de AWS.

Pruebas	Detalle a entregar
template.yaml	Detalle de integración con Serverless Framework Completar en documento final a entregar: Plantilla Solución CP1.docx
Bloque código pruebas unitarias	Detalle pruebas realizadas y localización en repositorio Completar en documento final a entregar: Plantilla Solución CP1.docx
Bloque código pruebas integración	Detalle pruebas realizadas y localización en repositorio Completar en documento final a entregar: Plantilla Solución CP1.docx
Bloque código pruebas calidad	Detalle pruebas realizadas y localización en repositorio Completar en documento final a entregar: Plantilla Solución CP1.docx
Bloque código pruebas unitarias complejidad	Detalle pruebas realizadas y localización en repositorio Completar en documento final a entregar: Plantilla Solución CP1.docx
Bloque código pruebas unitarias seguridad	Detalle pruebas realizadas y localización en repositorio Completar en documento final a entregar: Plantilla Solución CP1.docx

Material a entregar

Rellenar completamente el documento que lleva por nombre **Plantilla Solución CP1.docx**, con las secciones requeridas para los supuestos de los apartados A y B (y C en caso de hacerlo), de los cuales se piden evidencias (tablas, capturas de pantalla, logs, fragmentos de código, etc.) que reflejen el correcto progreso del alumno en el despliegue de los *pipelines* de CI/CD para ambas aproximaciones o frameworks de operativización.

Nota: para su entrega, dicho documento de plantilla ha de exportar como PDF.

En la plantilla de la solución se ha de incorporar el enlace al repositorio de código del alumno con el código fuente como propuesta de la solución.

Anexo I. Entornos de ejecución

Ejecución en entorno Legacy/On premise/Clásico

Entorno de ejecución clásico donde se dispone de un servidor físico. Este servidor físico tiene una asignación de recursos específica, con su configuración, tanto a nivel de Hardware como de Software particular en función de las diferentes aplicaciones que se quieran desarrollar. Los problemas típicos de esta aproximación son:

- ▶ Ante la falta de recursos se ha de acometer la compra de nuevos recursos Hardware.
- ▶ El control de la infraestructura a nivel de almacenamiento de datos y copias de seguridad.
- ▶ Los planes de Disaster Recovery suelen ser muy costosos y complejos.
- ▶ Se requieren especialistas en sistemas, que se responsabilicen del mantenimiento del Software (SSOO, drivers, software de HW utilizado, etc.) y Hardware (Cabinas de discos, firmwares, redes...).
- ▶ Necesidad de segregar los entornos físicos -> DEV, INT, UAT, QA, PRE y PRO.
- ▶ Problemas de las librerías incorporadas o las dependencias directas (implementaciones y versiones).
- ▶ Escalado Horizontal.
- ▶ Monitorización global.

A partir de este tipo de entorno de ejecución, existen una serie de refinamientos sobre las soluciones Legacy, muy especializadas según el contexto, como pueden ser:

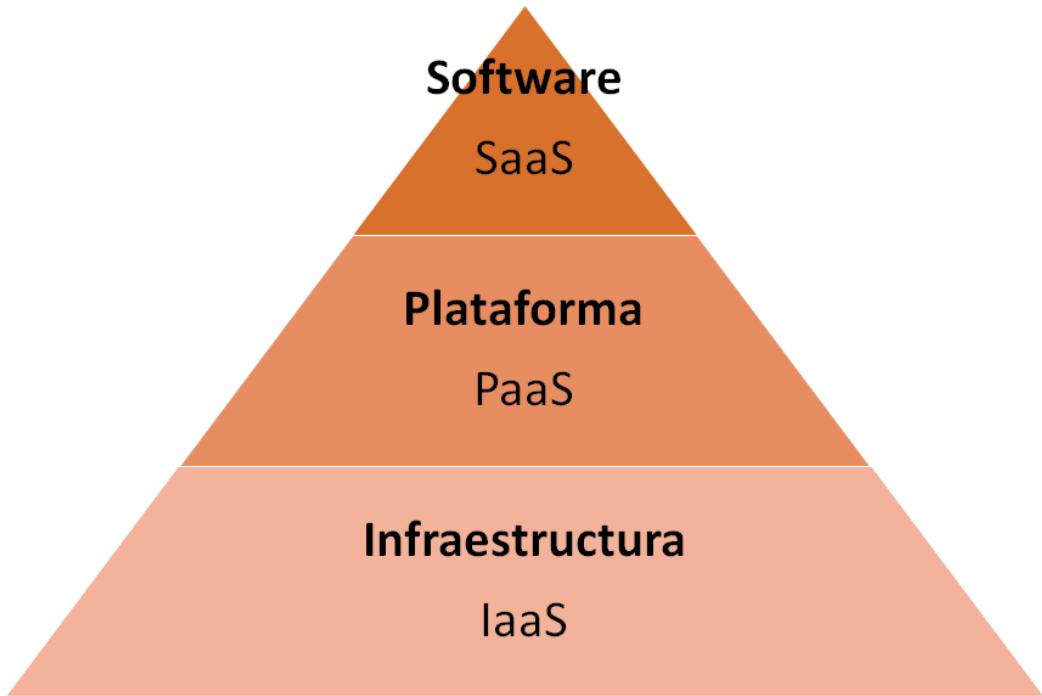
- ▶ Uso de Mainframes -> máquinas con recursos HW impensables para un usuario medio, muy usadas en entornos financieros, por su robustez en el manejo de transacciones.

- ▶ Mejoras en las infraestructuras de comunicaciones -> Internet, redes LAN, redes móviles, etc.
- ▶ Aparición de las máquinas virtuales y entornos virtualizados -> como infraestructuras basadas en VMWare, Citrix u Openstack.
- ▶ Aparición de la virtualización bajo demanda -> Facilita la creación y configuración de entornos virtualizados de forma automática (Por ejemplo: Vagrant, Heroku, etc.).
- ▶ Aparición de automatización de infraestructuras -> Facilita automatizar los procesos de configuración de los servidores (Por ejemplo: Puppet, Chef, Ansible, etc.).
- ▶ Infraestructura como código (IaaS) -> refinamiento máximo del despliegue de una infraestructura, a través de las APIs de los diferentes proveedores de servicios Cloud, tanto públicos como privados.
- ▶ Aparición de los contenedores (por ejemplo: Docker, Rocket, etc.).

Ejecución en entorno Cloud

Desde hace más de 10 años se ha estado produciendo un cambio de enfoque en muchas empresas en lo que se refiere al aprovisionamiento de infraestructura, todo esto gracias a la aparición de los entornos de ejecución *Cloud*. Un entorno *Cloud* se refiere a un recurso hardware, como por ejemplo un servidor físico, proporcionado por un proveedor (Azure, AWS, Google, etc.). Esta asignación de recursos puede ser específica o dinámica, y habilita configuraciones particulares de los lenguajes/versiones a utilizar donde se ejecutan las diferentes aplicaciones. Además, estos recursos físicos se encuentran repartidos por los diferentes CPD de los proveedores, por lo que los clientes de los servicios Cloud no pueden verlos físicamente, pero sí trabajar con ellos como si se encontraran bajo su control.

Dentro del modelo de *Cloud* existen una serie de modelos de servicio, a la hora de realizar la distribución de SW. Estos modelos de servicios son:



- ▶ **IaaS (Infraestructura como Servicio).** Modelo basado en que un proveedor se encarga de la distribución de la infraestructura necesaria:
 - También se denomina HaaS (Hardware como Servicio):
 - Unidad de despliegue: el sistema operativo.
 - Objetivo: externalizar el servidor (espacio disco, tiempo de computación y/o base de datos).
 - Proporciona: proporciona máquinas virtuales empaquetadas con sistemas operativos.
 - Abstracción: servidores físicos.
 - Características: escalabilidad, elasticidad, disponibilidad, seguridad, automatización, mantenibilidad, etc.
 - Pago: por configuración y uso.
 - Ejemplo AWS: EC2, RDS.

- ▶ **PaaS (Plataforma como Servicio).** Modelo basado en que un proveedor se encarga de proporcionar TODO lo necesario para soportar un ciclo de vida completo de puesta en marcha de aplicaciones y servidores:
 - Unidad de despliegue: las aplicaciones.

- Objetivo: externalizar la aplicación (base de datos, SSOO, servidor de aplicaciones).
 - Proporciona: proporciona plataformas de aplicaciones para el desarrollo.
 - Abstracción: sistemas operativos y middleware.
 - Características: lo mismo que IaaS.
 - Pago: por licenciamiento, configuración y uso.
 - Ejemplos: Kubernetes, Openshift, AWS Lightsail.
- **SaaS (Software como servicio):** Modelo basado en que un proveedor se encarga de proporcionar mantenimiento, soporte y operación a un cliente durante un periodo de contratación de servicio:
- Unidad de despliegue: los servicios.
 - Objetivo: externalizar el uso.
 - Proporciona: proporciona servicios/funcionalidad bajo demanda.
 - Abstracción: aplicaciones online.
 - Características: lo mismo que IaaS.
 - Pago: por volumen de usuarios, módulos utilizados, plan de soporte, acceso por dispositivo.
 - Ejemplos: AWS Translate, AWS Poly.

Ejecución en entorno Infraestructura como Código

Este entorno surge como evolución «natural» del entorno de ejecución *Cloud* es decir desde que aparece la posibilidad de pagar por el uso y la creación de entornos casi automática y con tareas de mantenimiento que pueden variar según la habilidad del técnico y las capacidades de las herramientas y API's con las que se trabaje.

Ahora lo importante es automatizar el proceso para NO tener que repetir pasos manualmente cada vez que se crea un entorno (por ejemplo: instalar Linux con un Apache Tomcat y la JDK 8). Y además porque como existen en ocasiones

dependencias a ciertas versiones de elementos, se ha de ser capaz de generar clones en ciertos contextos.

Las características que son cubiertas con este enfoque son:

- ▶ Creación de una infraestructura versionable, reproducible, consistente e independiente del entorno.
- ▶ Disponer de entornos limpios y sin estados anteriores.
- ▶ Automatización del proceso manual mediante una explotación programática sobre los entornos de ejecución: clásico y Cloud.
- ▶ Facilita la creación / destrucción de entornos como ciclo de integración / entrega continua.
- ▶ Uso de herramientas específicas.

Ejecución en entorno Serverless

Este entorno surge a partir de la existencia del entorno de ejecución «Cloud» con modelo de servicio SaaS en concreto del enfoque FaaS (Function as a Service).

Un entorno de ejecución Serverless es un servicio proporcionado por un proveedor (Azure, AWS, Google, etc.) donde se facilita un servidor (físicos o *Cloud*), la asignación dinámica de sus recursos, una configuración concreta de los lenguajes/versiones a utilizar y una única función de entrada como contrato.

En líneas generales se consigue:

- ▶ No existe aprovisionamiento, gestión y mantenimiento de servidores.
 - Toda la infraestructura está delegada al proveedor.

- ▶ Enfoque On-Demand / Bajo demanda:
 - Funcionamiento basado en el uso.
 - Facturación basada en el uso -> Sólo se paga por los recursos que se consumen.
 - Como los recursos los establece el proveedor la medida de facturación suele estar asociada con el tiempo de ejecución) -> "más tiempo es más coste".
 - Escalado continuo/elástico basado en el uso.
 - Reaprovechamiento de contenedores.
- ▶ Disponibilidad y Tolerancia a fallos (por defecto suele ser proporcionado por el proveedor) -> Reduce las preocupaciones de los desarrolladores.
 - Ante un error el proveedor cambia el código de máquina sin que el usuario se entere.
- ▶ Facilita la orientación a Eventos -> Arquitecturas EDA (Event Driven Architecture)
-> Buen comportamiento.
- ▶ Latencia establecida por el proveedor
 - Un evento invoca a esta función que genera el entorno y una vez ejecutado, ese entorno desaparece -> arranque “frío” / “caliente” -> latencia.
- ▶ Ahorro de costes en general si se hace un buen uso y se utiliza para lo que realmente fue diseñado.

Anexo II . Referencias

Amazon Web Services. URL: <https://aws.amazon.com/es/>

Amazon Cloud9. URL: <https://aws.amazon.com/es/cloud9/>

Amazon CloudWatch. URL: <https://aws.amazon.com/cloudwatch/>

Amazon CodeCommit. URL: <https://aws.amazon.com/codecommit/>

Amazon CodeBuild. URL: <https://aws.amazon.com/codebuild/>

Amazon CodeDeploy. URL: <https://aws.amazon.com/es/codedeploy/>

Amazon CodePipeline. URL: <https://aws.amazon.com/codepipeline/>

Amazon API-Gateway. URL: <https://aws.amazon.com/es/api-gateway/>

Amazon Elastic Cloud Computing. URL: <https://aws.amazon.com/es/ec2/>

Amazon Lambda. URL: <https://aws.amazon.com/es/lambda/>

Amazon Simple Storage Service. URL: <https://aws.amazon.com/es/s3>

Amazon DynamoDB. URL: <https://aws.amazon.com/es/dynamodb>

Amazon CloudFormation. URL: <https://aws.amazon.com/es/cloudformation/>

Amazon IAM. URL: <https://aws.amazon.com/es/iam/>

AWS Services Supported with AWS Academy Starter Account. URL:

https://awsAcademy-starter-account-services.s3.amazonaws.com/AWS_Academy_Starter_Account_Services_Supported.pdf

Security Best Practices in IAM. URL:

<https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html>

CodeCommit Userguide. URL:

<https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html>

Individual user setup for AWS Cloud9 - AWS Cloud9. URL:

<https://docs.aws.amazon.com/cloud9/latest/user-guide/setup-express.html>

AWS Cloud9 - User Guide. URL: <https://docs.aws.amazon.com/cloud9/latest/user-guide/aws-cloud9-ug.pdf>

AWS Serverless Application Model. URL:

<https://aws.amazon.com/es/serverless/sam/>

AWS Rest with DynamoDB. URL:

<https://github.com/serverless/examples/tree/master/aws-Python-rest-api-with-dynamodb>

To Do List. URL: https://satisfactory.gamepedia.com/To_Do_List

Github. URL: <https://github.com/>

Repositorio base de la parte 1. URL: <https://github.com/rgaleanog/todo-list-aws>

Amazon Comprehend - Developer Guide. URL:

<https://docs.aws.amazon.com/comprehend/latest/dg/comprehend-dg.pdf>

Amazon Translate - Guía para desarrolladores. URL:

https://docs.aws.amazon.com/es_es/translate/latest/dg/translate-dg.pdf

Pytest. URL: <https://pytest.org/>

Moto Library URL: <https://github.com/spulec/moto>

Localstack Library. URL: <https://github.com/localstack/serverless-localstack>

Pynamo Library. URL: <https://pynamodb.readthedocs.io/en/latest/index.html>

Coverage Library. URL: <https://coverage.readthedocs.io/en/coverage-5.2.1/>

Gatling plugin Library. URL: <https://github.com/gatling/gatling-maven-plugin-demo>

AWS Cli Reference - Create CodeCommit Repository. URL:

<https://awscli.amazonaws.com/v2/documentation/api/latest/reference/codecommit/create-repository.html>

Crear un repositorio de AWS CodeCommit. URL:

https://docs.aws.amazon.com/es_es/codecommit/latest/userguide/how-to-create-repository.html

Migrate a Git repository to AWS CodeCommit. URL:

<https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-migrate-existing.html>

Scaffolding. URL: [https://en.wikipedia.org/wiki/Scaffold_\(programming\)](https://en.wikipedia.org/wiki/Scaffold_(programming))

AWS Serverless Application Model (SAM) Library. URL:

<https://github.com/aws/serverless-application-model/blob/master/versions/2016-10-31.md>

AWS SAM template for a DynamoDB application. URL:

<https://docs.aws.amazon.com/lambda/latest/dg/kinesis-tutorial-spec.html>

Sam local invoke. URL: https://docs.aws.amazon.com/en_pv/serverless-application-model/latest/developerguide/sam-cli-command-reference-sam-local-invoke.html

Build specification reference for CodeBuild. URL:

<https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html>

Draw.io. URL: <https://app.diagrams.net/>