



Experto Universitario en DevOps & Cloud

Caso Práctico 1

Apartado C

Guía de apoyo

Índice

Introducción	3
Herramientas requeridas	4
Reto 1 – Clonado del repositorio	5
Reto 2 – Infraestructura AWS	6
Reto 3 – Stack Serverless	14
Reto 4 – Verificación API	17

Introducción

Este documento sirve de ayuda para la realización del apartado C del CP1, explicando, con mayor detalle, el enfoque a realizar para finalizar todos los ejercicios. En este apartado C, esta guía sí va a proporcionar instrucciones paso a paso, especialmente en los aspectos más complejos y técnicos, donde se requiere un conocimiento muy específico y avanzado del manejo de AWS.

No obstante, en ningún caso van a ser tareas vacías de retos, porque las instrucciones serán genéricas en otros aspectos, donde el alumno tendrá que buscar las soluciones. En este apartado, es bastante común cometer errores en alguno de los pasos, lo cual puede llegar a corromper totalmente el trabajo realizado hasta el momento, por lo que se recuerda al alumno la importancia de asegurarse bien de lo que se está haciendo en cada momento.

En este Apartado C, contamos con 4 retos:

- ▶ Clonado del repositorio de código del nuevo proyecto (To-Do's list): 10%
- ▶ Creación de toda la infraestructura AWS para llevar a cabo esta práctica: 40%
- ▶ Automatización del stack Serverless para el proyecto de To-Do's: 30%
- ▶ Comprobación de que la API Rest está funcionando correctamente: 20%

Es importante recordar que la calificación de cualquiera de los retos del CP1 se realiza en base a tres criterios:

- ▶ Funcionamiento correcto de la solución
- ▶ Calidad técnica de la implementación
- ▶ Explicación de cómo se ha llevado a cabo la implementación, demostrando conocimiento suficiente de la materia.

Herramientas requeridas

Las herramientas que se requieren para este apartado son:

- ▶ JDK (para ejecutar Jenkins)
 - Cada producto puede requerir una versión diferente, por lo que hay que elegir la que sea necesaria según las instrucciones de instalación de Jenkins.
- ▶ Jenkins (última versión estable)
 - Es importante usar la configuración por defecto, con los plugins por defecto, para mantener todos los alumnos la misma versión aproximada.
- ▶ Cloud9
 - Usaremos una AMI que ya está preparada para Cloud9, por lo que el proceso será más sencillo.
- ▶ Git
 - También disponible en la AMI que usaremos.
- ▶ Otros recursos de AWS para el despliegue de las herramientas necesarias para la práctica.

JDK: entorno de desarrollo Java y máquina virtual. Se puede descargar desde la página oficial de Oracle o bien mediante los instaladores habituales de cada S.O. (p.e. “apt-get”).

Usaremos la versión que nos recomiende Jenkins.

Jenkins: se instala fácilmente siguiendo todos los pasos indicados en el instalador.

Para este apartado C, usaremos la versión Ubuntu/Debian, por lo que deben seguirse las instrucciones de instalación que el fabricante indica para esta distribución.

Sistema Operativo: usaremos Linux Ubuntu tanto para el apartado C como para posteriormente el D.

Reto 1 – Clonado del repositorio

A partir del apartado C, ya dejamos atrás la aplicación de la calculadora y, desde este momento, usaremos un proyecto distinto, que realiza una gestión de tareas (To-Do's), con persistencia en una base de datos no-SQL.

Este proyecto se encuentra alojado en el siguiente repositorio de GitHub:

- <https://github.com/anieto-unir/todo-list-aws>

El primer reto de este apartado C consiste en que el alumno dé de alta un repositorio en su cuenta de GitHub, con el mismo código fuente de este repositorio.

En ningún caso se permitirá que el alumno trabaje directamente contra el repositorio original (salvo para descargar el código fuente la primera vez).

A continuación, se detallan las tareas que deben llevarse a cabo en este primer reto:

- ▶ Descarga del código fuente, y creación del repositorio del alumno con este mismo código fuente.
 - Indicar la URL del nuevo repositorio
 - Lista de comandos ejecutados y pasos manuales en GitHub, para llevar a cabo la creación del nuevo repositorio.

Este repositorio puede contener código o ficheros de configuración que no se usarán para nada en esta práctica. Se recuerda igualmente que los pipelines que aparecen en el repositorio no se usan tampoco para nada ni sirve de ayuda para los que tendremos que crear en el apartado D.

Reto 2 – Infraestructura AWS

Este reto es el más complejo del apartado C, puesto que es muy fácil equivocarse, o no realizar de forma 100% correcta los pasos a ejecutar, por lo que se recuerda nuevamente mucho cuidado al dar cada paso, y es aquí donde se pueden dilatar los tiempos de ejecución de la práctica.

Algunos de las capturas de pantalla que se ofrecerán en esta guía, pueden no ser iguales a las herramientas de gestión de AWS, puesto que estas herramientas evolucionan constantemente, y la guía no puede reflejar la UI actualizada en cada momento. No obstante, sí ofrecerán datos suficientes como para llevar a cabo los pasos sin ningún problema. Además, (casi) todos estos pasos se han visto en clase.

En el apartado de entregables de este reto, el alumno deberá indicar los comandos y acciones realizados en todo momento, mediante explicación y captura de pantalla.

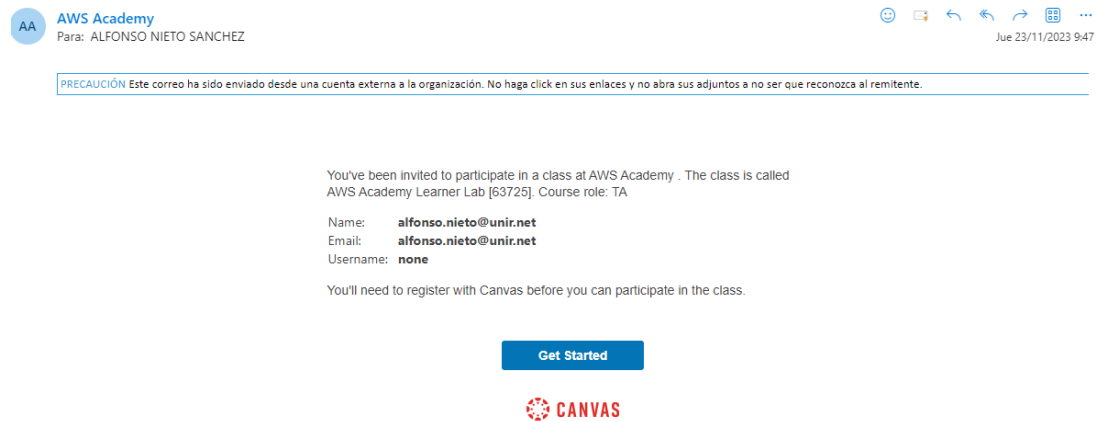
Las tareas a realizar para montar la infraestructura son las siguientes:

- ▶ Acceso al laboratorio
- ▶ Creación instancia EC2
- ▶ Instalación Cloud9
- ▶ Instalación Jenkins

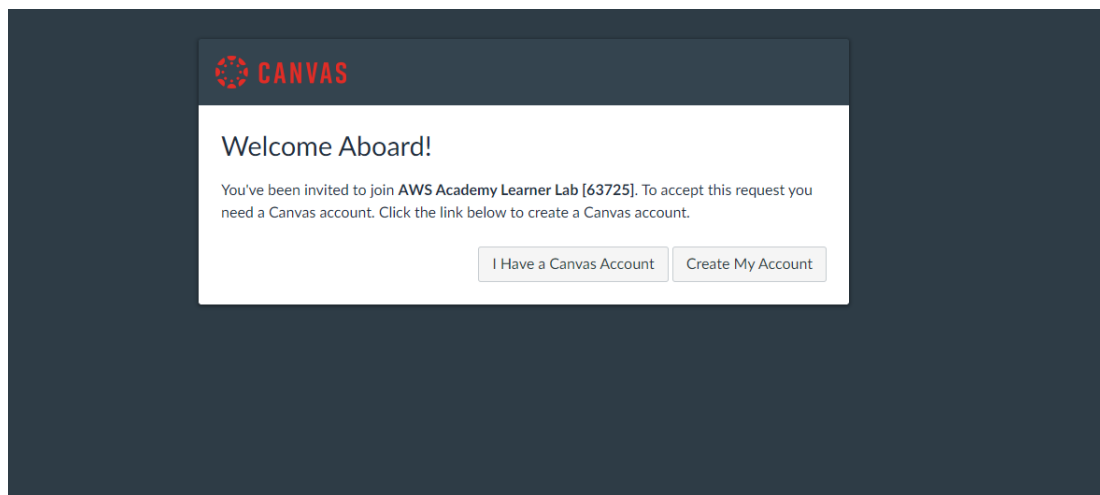
A continuación se detallan estas 4 tareas.

- ▶ Acceso al laboratorio
 - A partir del correo de AWS Academy que cada alumno debe haber recibido, debe pulsarse en “Get Started”, y seguir las instrucciones para dar de alta la cuenta de Canvas (si no se tiene ya) y finalmente acceder a la página principal de AWS Academy.

Correo:



Creación de usuario Canvas:

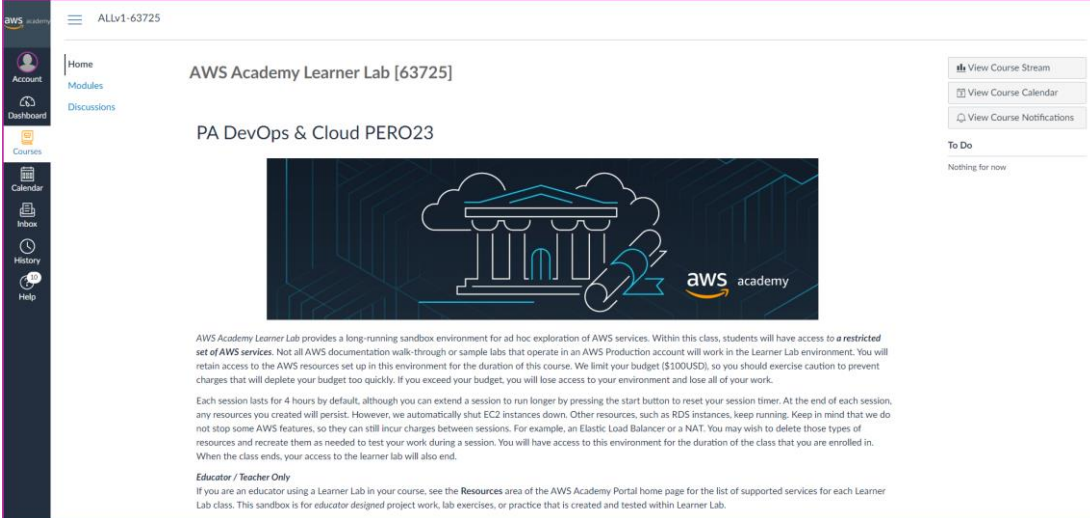


Identificación del usuario (esta pantalla puede ser diferente para los alumnos):



Una vez elegido “Student Login”, se muestra el formulario de usuario/contraseña.


Página principal de AWS Academy:



Home Modules Discussions

AWS Academy Learner Lab [63725]

PA DevOps & Cloud PERO23



AWS Academy Learner Lab 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 Learner Lab environment. You will retain access to the AWS resources set up in this environment for the duration of this course. We limit your budget (\$100USD), 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.

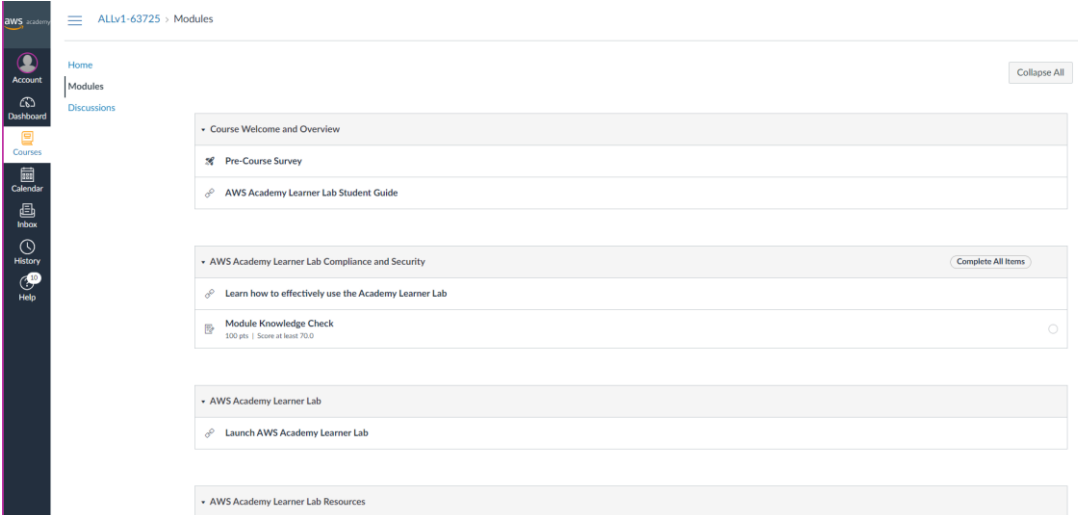
Each session lasts for 4 hours by default, although you can extend a session to run longer by pressing the start button to reset your session timer. At the end of each session, any resources you created will persist. However, we automatically shut EC2 instances down. Other resources, such as RDS instances, 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 that you are enrolled in. When the class ends, your access to the learner lab will also end.

Educator / Teacher Only
If you are an educator using a Learner Lab in your course, see the **Resources** area of the AWS Academy Portal home page for the list of supported services for each Learner Lab class. This sandbox is for educator designed project work, lab exercises, or practice that is created and tested within Learner Lab.

View Course Stream
View Course Calendar
View Course Notifications

To Do
Nothing for now

Página “Modules”:



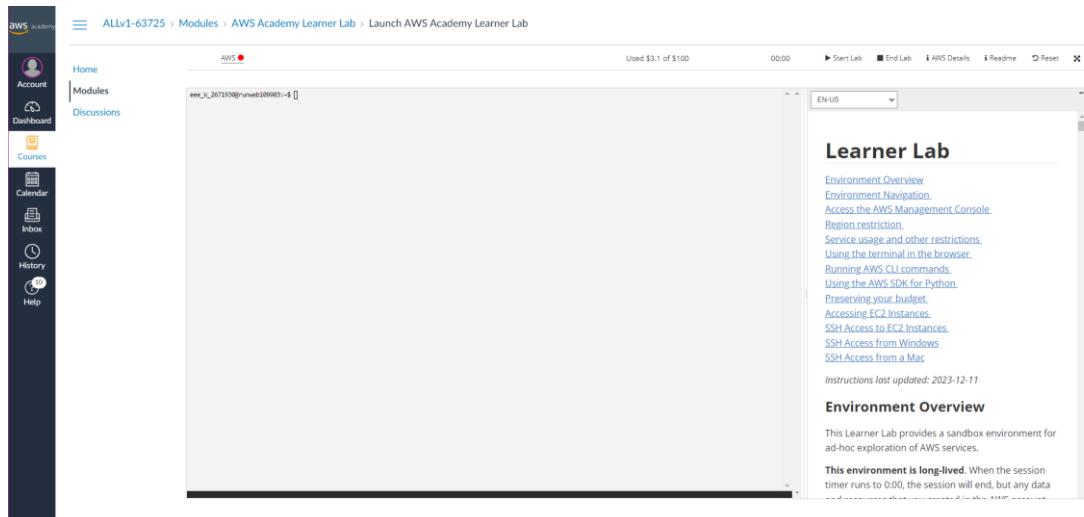
Home Modules Discussions

AWS Academy Modules

Collapse All

- Course Welcome and Overview
 - Pre-Course Survey
 - AWS Academy Learner Lab Student Guide
- AWS Academy Learner Lab Compliance and Security Complete All Items
 - Learn how to effectively use the Academy Learner Lab
 - Module Knowledge Check
100 pts | Score at least 70.0
- AWS Academy Learner Lab
 - Launch AWS Academy Learner Lab
- AWS Academy Learner Lab Resources

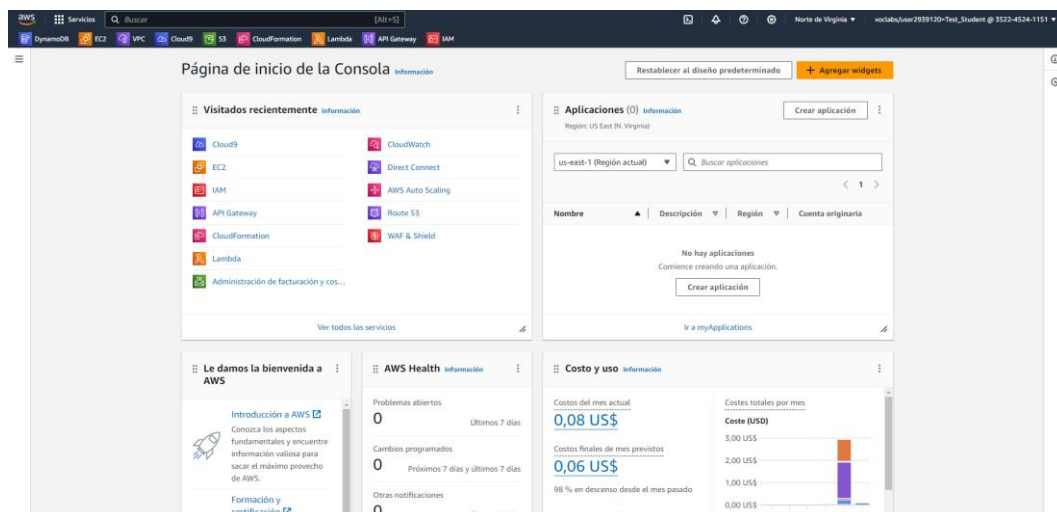
- Una vez en esta página previa al acceso al laboratorio, se recuerda al alumno que se dispone de dos documentos que explican el funcionamiento de AWS Academy y que son de lectura recomendada para el alumno. En el caso del documento “AWS Academy Learner Lab Student Guide”, el documento es de obligatoria lectura, puesto que describe con detalle el funcionamiento de la plataforma desde un punto de vista práctico.
- Una vez leída la documentación, se accede al panel de control de AWS Academy pulsando en el enlace “Launch AWS Academy Learner Lab”.



Este panel de control nos indica, con el punto rojo, junto al símbolo AWS, que el laboratorio está parado en este momento.

Se recuerda al alumno que la sesión del laboratorio dura 4 horas y que finalizará tras ese tiempo. El alumno puede volver a iniciarla (Start Lab), pero lo más importante es que, una vez finalice cada sesión de trabajo, el alumno pare el laboratorio (End Lab), sin esperar a que automáticamente se cierre tras las 4 horas, para dejar de consumir créditos.

Pulsando el botón-enlace AWS, que una vez iniciado el laboratorio, nos aparecerá con el punto en verde, accedemos a la consola de administración de AWS, donde podemos revisar todos los recursos que hemos creado, así como desplegar nuevos recursos.



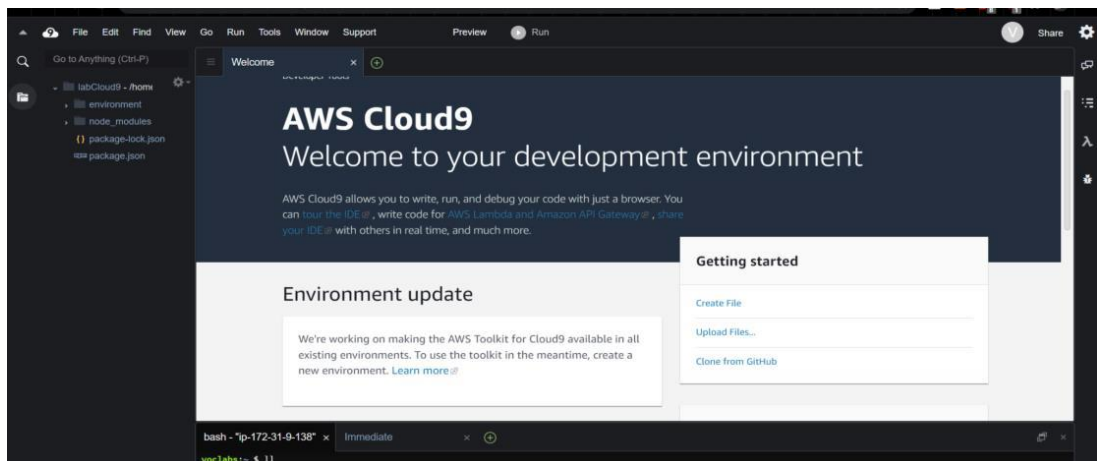
► Creación instancia EC2

Una vez familiarizados con el entorno de la consola (podemos elegir servicios favoritos), echar un vistazo a todos los servicios, etc., comenzaremos la creación de nuestra instancia EC2, que contendrá Jenkins y Cloud9, en un entorno Ubuntu.

Para esta etapa, no se ofrecerán capturas de pantalla, sino pasos a realizar. Los datos aquí facilitados no tienen por qué seguir el mismo orden a como se piden en el formulario de creación de la EC2:

- Creación instancia EC2 a partir de una AMI
 - Buscaremos una AMI (Amazon Machine Image) mediante el buscador de AMIs, filtrando por “Imágenes Públicas” (“Public Images”), también por Alias de Propietario “Amazon” y finalmente por Origen “amazon/Cloud9Ubuntu22”.
 - Del listado de imágenes, elegiremos la más reciente y lanzaremos la instancia EC2 a partir de esta imagen, usando el botón que se nos muestra claramente en pantalla.
 - Debemos elegir el dimensionamiento de la máquina, que será una T3.micro (o T3.small)
 - El Rol con el que se ejecutará esta máquina será el LabInstanceProfile.
 - El almacenamiento a provisionar será un sistema de bloques de 20GB, de tipo gp2 (es el que se muestra por defecto actualmente).
 - A nivel de seguridad de red, debemos habilitar acceso SSH desde cualquier IP en internet, así como acceso TCP a los puertos 80 y 8080 (tráfico HTTP). A elección del alumno, podrá restringirse el acceso a estos puertos desde la IP de la red local del alumno, o bien dejarlo abierto a internet (inseguro).
 - El par de claves a usar será la que se llama “vockey”.
- Acceder a la instancia EC2 mediante SSH, desde la terminal que nos proporciona el propio laboratorio.
 - Desde la página de EC2, acceder a la información de la instancia EC2 recién creada, y obtener su dirección IP/DNS públicos (temporal)
 - Acceder mediante SSH a la IP/DNS públicos, usando como credenciales la clave digital que tenemos en la propia terminal:
 - `ssh -i ~/.ssh/labsuser.pem ubuntu@public-ip`

- Verificar con los comando habituales que la máquina es tal y como la hemos configurado (S.O., almacenamiento, etc.): `uname`, `df`, etc.
- ▶ Integración con Cloud9
 - Para la integración de Cloud9, usaremos el asistente que nos ofrece la consola administrativa de AWS, en el apartado de Cloud9.
 - Pulsaremos en Crear Entorno.
 - Nombre y descripción serán “labCloud9” (puede ser cualquier nombre)
 - Indicaremos que no queremos que se cree una instancia EC2 nueva, sino que queremos aprovechar la instancia EC2 que hemos desplegado en la tarea anterior.
 - Seleccionaremos usuario Ubuntu, el host de la instancia EC2 (IP pública o DNS público)
 - Finalmente tenemos que autorizar, en nuestra instancia EC2, a que Cloud9 pueda conectarse por SSH a la instancia, para lo cual debemos copiar la clave de Cloud9 en nuestro fichero de claves autorizadas de la máquina EC2.
 - Dentro de la máquina EC2, a la cual hemos accedido por SSH en el punto anterior, editaremos con cualquier editor de texto, el fichero “.ssh/authorized_keys”. Por ejemplo, podemos usar `vi`, `vim`, `nano`, etc.
 - En este fichero debemos añadir, al final del mismo, la clave que hemos copiado en el asistente de Cloud9, al pulsar en el botón “Copy key to clipboard” o “Copiar la clave al portapapeles” (según idioma de la consola).
 - Finalmente revisamos que todo esté bien y confirmamos la creación del entorno.
 - Si todo ha ido bien, nos ofrecerá un enlace para acceder a nuestro recién creado entorno de Cloud9, que se ejecuta en la máquina EC2.
 - Una vez abierto, deberíamos ver algo similar a esto:



► Instalación de Jenkins

- La instalación de Jenkins la podemos realizar bien desde el cliente SSH que hemos abierto desde la terminal de AWS Academy, o bien desde la terminal que ahora nos ofrece Cloud9, en la parte inferior de la pantalla.
 - Se recomienda usar el terminal de Cloud9, por ser más sencillo de usar y más completo visualmente.
 - Los comandos para instalar Jenkins están indicados en la web del fabricante, por lo que se remite al alumno a que siga los pasos que se indican en la web, en el apartado de descargas, teniendo en cuenta el sistema operativo que estamos usando (Linux Ubuntu/Debian)
 - Una vez finalizada la instalación de Jenkins y sus requisitos (JDK), validaremos que el servicio de Jenkins está arrancado, mediante el comando:
 - `sudo service Jenkins status`
 - Ahora que tenemos Jenkins en marcha, comprobaremos que podemos acceder a Jenkins vía web, en el puerto 8080, mediante la IP o DNS públicos de la instancia EC2, que es donde corre Jenkins.
 - Una vez accedamos, completaremos la configuración de Jenkins hasta que finalmente accedamos a la pantalla principal de Jenkins donde podríamos crear pipelines, acceder a la configuración, etc. (cosa que no haremos en este apartado C).
- Creación de una IP pública estática (en AWS se llama IP elástica)
- Por defecto, la iniciar una instancia EC2, se nos asigna una IP pública y un dominio público (reconocible en internet). Sin embargo, cada vez que

apagamos el laboratorio (y, por tanto, se apagan las máquinas virtuales), esta IP se pierde, y se asignará una nueva en la siguiente sesión de trabajo.

- Debemos proporcionar un mecanismo que nos permita poder acceder siempre a nuestra instancia EC2 desde una IP Pública fija.
- Queda como tarea del alumno, investigar cómo podemos conseguir este objetivo.
- Hay que tener en cuenta que hemos configurado Cloud9 para que se conecte a nuestra instancia EC2 por lo que, si hemos configurado Cloud9 con la IP no fija de la instancia EC2, cuando iniciemos de nuevo el laboratorio, Cloud9 no funcionará puesto que no será capaz de conectar con la instancia EC2 que le indicamos inicialmente.

En este punto, habremos acabado este segundo reto del apartado C.

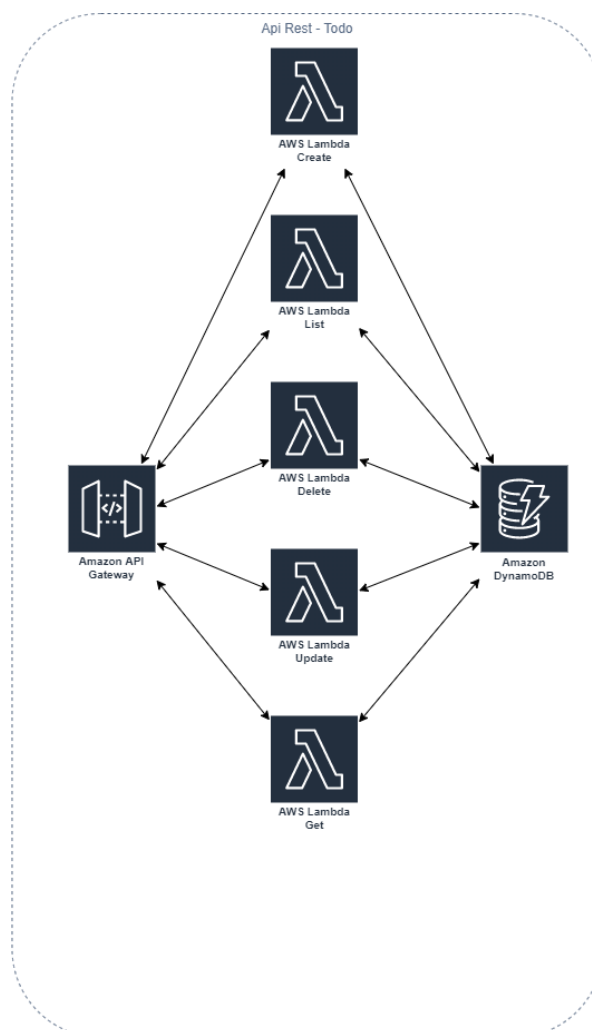
Como entregables, el alumno debería indicar pantallazos y comandos empleados en cada uno de los pasos (se pueden obviar pasos superfluos que no aporten valor).

Reto 3 – Stack Serverless

Nuestra instancia EC2 únicamente contiene Jenkins y un entorno de desarrollo para facilitarnos el trabajo, en cuanto a que nos ofrece una interfaz gráfica productiva, y unos comandos ya preinstalados muy útiles para los siguientes pasos.

Sin embargo, el objetivo no es levantar nuestra API Rest en la instancia EC2, sino que debemos hacer uso de los recursos Serverless para lograr este fin, puesto que estos servicios tienen un coste muy bajo, se crean fácilmente, y ofrecen grandes ventajas en cuanto a autodimensionamiento, alta disponibilidad y conectividad.

Partimos de una plantilla SAM donde está ya realizada la configuración de todo el **stack** que queremos levantar, y que se puede apreciar en la siguiente imagen.



Este Stack está formado por un API Gateway, 5 funciones lambda, y una tabla de DynamoDB.

El alumno debe familiarizarse con la plantilla, para entender cómo se realiza la definición de este stack.

Las tareas a realizar son:

- ▶ Descarga del repositorio en una carpeta todo-list-aws
 - Usaremos git desde la terminal de Cloud9
 - Verificaremos que nos aparece, en el panel izquierda de Cloud9, la nueva carpeta que se ha creado y que contiene todos los ficheros del repositorio.
 - Accederemos a la carpeta todo-list-aws donde hemos descargado todo el código fuente.
- ▶ Construcción de la plantilla SAM, mediante los comandos
 - `sam build`
- ▶ Despliegue en AWS de este stack, mediante el comando
 - `sam deploy --guided`
 - En el asistente de despliegue, usaremos los siguientes parámetros:

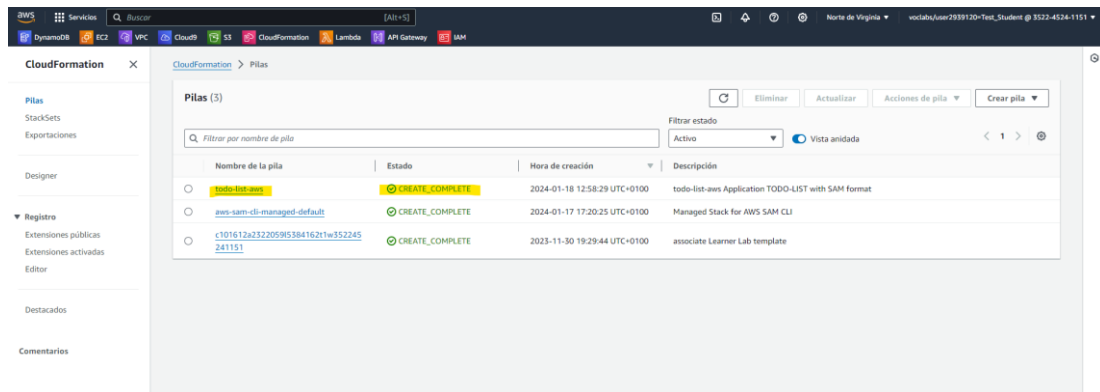
```
voclabs:~/todo-list-aws (master) $ sam deploy --guided

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]:
AWS Region [us-east-1]:
Parameter Stage [default]: staging
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
Confirm changes before deploy [Y/n]:
#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]:
#Preserves the state of previously provisioned resources when an operation fails
Disable rollback [Y/n]:
CreateTodoFunction has no authentication. Is this okay? [y/N]: y
ListTodosFunction has no authentication. Is this okay? [y/N]: y
GetTodoFunction has no authentication. Is this okay? [y/N]: y
UpdateTodoFunction has no authentication. Is this okay? [y/N]: y
DeleteTodoFunction has no authentication. Is this okay? [y/N]: y
Save arguments to configuration file [Y/n]:
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]: staging
```

- ▶ Revisión de los Outputs que ha generado el despliegue, con todas las URLs a los recursos que hemos creado (entradas en el API Gateway, 5 funciones lambda y 1 tabla DynamoDB, además de otros recursos automáticos adicionales para la gestión de permisos, etc.)
 - Debemos guardar esas URLs, porque las usaremos en el reto 4
- ▶ Comprobación de que el Stack se ha creado correctamente desde la sección de CloudFormation, en la consola administrativa de AWS.



Reto 4 – Verificación API

Ya hemos conseguido levantar nuestro Stack con la API Rest funcionando correctamente para permitir la gestión de tareas To-Do.

Bueno, o eso creíamos, como veremos más adelante.

En este último reto, vamos a verificar que nuestra API Rest atiende nuestras peticiones y almacena la lista de tareas en la base de datos.

Para llegar a cabo esta validación, invocaremos a la API mediante el comando *curl* de Linux, aunque se podría usar otro comando similar, o bien una herramienta tipo Postman, puesto que nuestro Stack está actualmente publicado en internet, accesible desde cualquier punto del planeta.

Si el alumno tiene conocimientos de programación de front-end, tampoco le será nada costoso preparar una UI HTML que acceda a este back-end API Rest para la gestión de tareas, pero eso queda fuera del alcance de este reto.

Nos centraremos en la validación de nuestra API mediante comandos *curl*, desde la propia consola de Cloud9, o desde nuestro PC local o desde donde queramos.

Al finalizar el reto anterior, se indicaba que importante guardar los datos de Outputs que nos devolvió *sam deploy*, que serán similares a estos:

```
CloudFormation outputs from deployed stack
-----
Outputs
-----
Key      BaseUrlApi
Description Base URL of API
Value    https://4nzj8zhwai.execute-api.us-east-1.amazonaws.com/Prod

Key      DeleteTodoApi
Description API Gateway endpoint URL for ${opt:stage} stage for Delete TODO
Value    https://4nzj8zhwai.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://4nzj8zhwai.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://4nzj8zhwai.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://4nzj8zhwai.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://4nzj8zhwai.execute-api.us-east-1.amazonaws.com/Prod/todos/
```

Esas son las URLs a las que deberemos acceder para invocar cada uno de los métodos de nuestra API Rest.

Se trata de una API Rest CRUD completa, con los métodos de Create, Retrieve (Get), Update, Delete y List.

Las tareas a realizar en este reto son:

- ▶ Ejecución de los comandos CURL para verificar toda nuestra API Rest:
 - Crear una nueva tarea To-Do:
 - `curl -X POST https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos -data '{ "text": "Learn Serverless" }'`
 - Debemos cambiar XXXXXXX, por el identificador de nuestro stack.
 - En el campo “text” podemos poner el valor que queramos, para componer una pequeña lista de tareas
 - Nos devolverá un texto json con los datos de la tarea que hemos creado, indicando su ID (importante, porque lo usaremos en los siguientes métodos), el texto de la tarea, y las fechas de creación y última actualización.
 - Listar todos los To-Do:
 - `curl https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos`
 - Obtener un To-Do a partir de su identificador:
 - `curl https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>`
 - Actualizar un To-Do existente, a partir de su identificador:
 - `curl -X PUT https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id> --data '{ "text": "Learn python and more", "checked": true }'`
 - Borrar un To-Do a partir de su identificador:
 - `curl -X DELETE https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>`
- ▶ El alumno deberá mostrar por pantalla la salida de todos estos comandos *curl*.
 - Efectivamente, todos los comandos van a devolver el mismo mensaje de error.
 - Hemos visto en clase cómo debe corregirse, por lo que aplicaremos el fix en el fichero “src/todoList.py” de nuestro proyecto, indicando el nombre correcto del recurso con el que queremos trabajar.
 - Antes de arreglar el fallo, debemos acceder a la consola administrativa de la función Lambda, realizar una prueba manual de invocación a la función, y

obtener los logs necesarios que nos permiten averiguar cuál era el problema que estaba ocurriendo.

- ▶ Una vez subsanado el error (y subido al repositorio de GitHub), debemos volver a construir nuestro Stack y desplegarlo, así como repetir las pruebas, donde se aprecie el resultado correcto, ahora sí, de las llamadas a *curl*.
- Quien desee usar Postman (o similar) para estas pruebas, puede usarlo igualmente, adjuntando capturas de las invocaciones a cada uno de los métodos.