Experto Universitario en DevOps & Cloud

Caso práctico 1. Enunciado

Índice

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 Serverless[1], 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.

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:

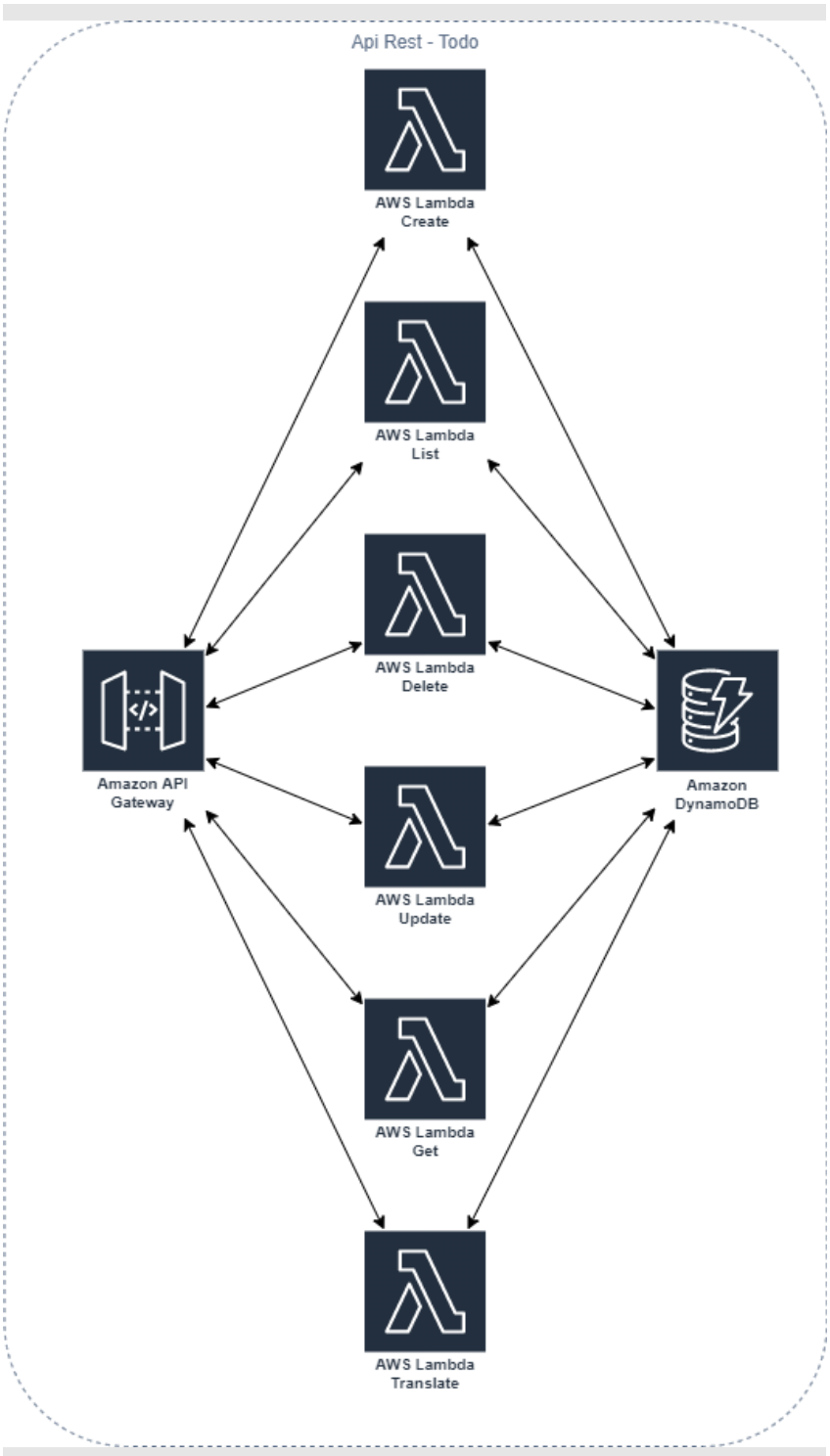
- SAM + Jenkins: El modelo de aplicaciones sin servidor (SAM) de AWS es un framework open source pensado para construir aplicaciones sin servidor. Proporciona una sintaxis abreviada para expresar funciones, API, bases de datos y mapeos de fuentes de eventos. Con solo unas pocas líneas por recurso, se 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, (con carácter optativo en el presente caso práctico), se requiere la ampliación de 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, los cuales tenéis disponibles 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.


Api Rest - Todo
AWS Lambda Create
AWS Lambda List
AWS Lambda Delete
Amazon API Gateway
Amazon DynamoDB
AWS Lambda Update
AWS Lambda Get
AWS Lambda Translate

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 *Pipeline*	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 continúa para automatizar los procesos de compilación de los artefactos software requeridos en el despliegue en el entorno productivo.

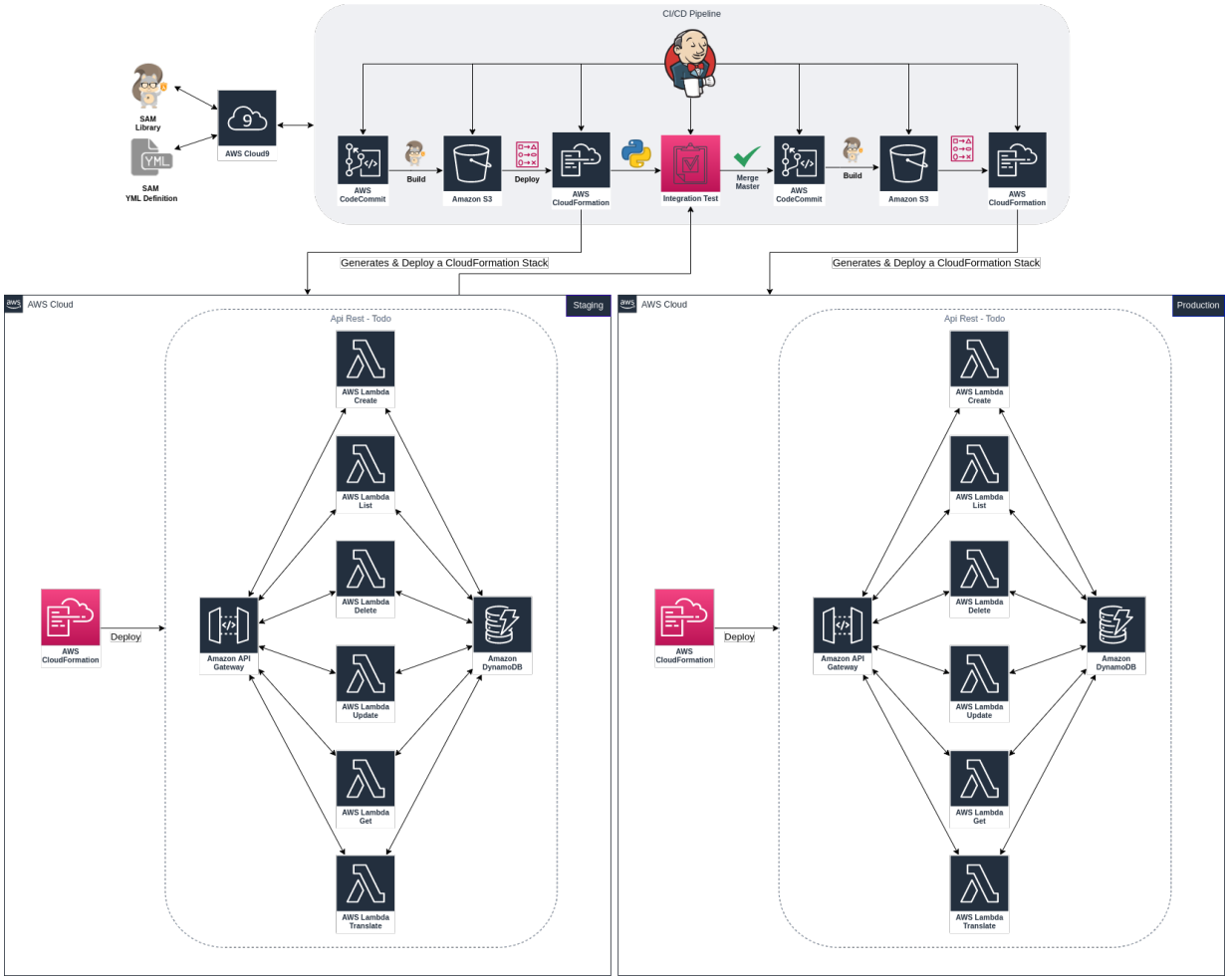
A continuación, se van a detallar brevemente cada uno de los servicios necesarios para la elaboración de este apartado:

- 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.

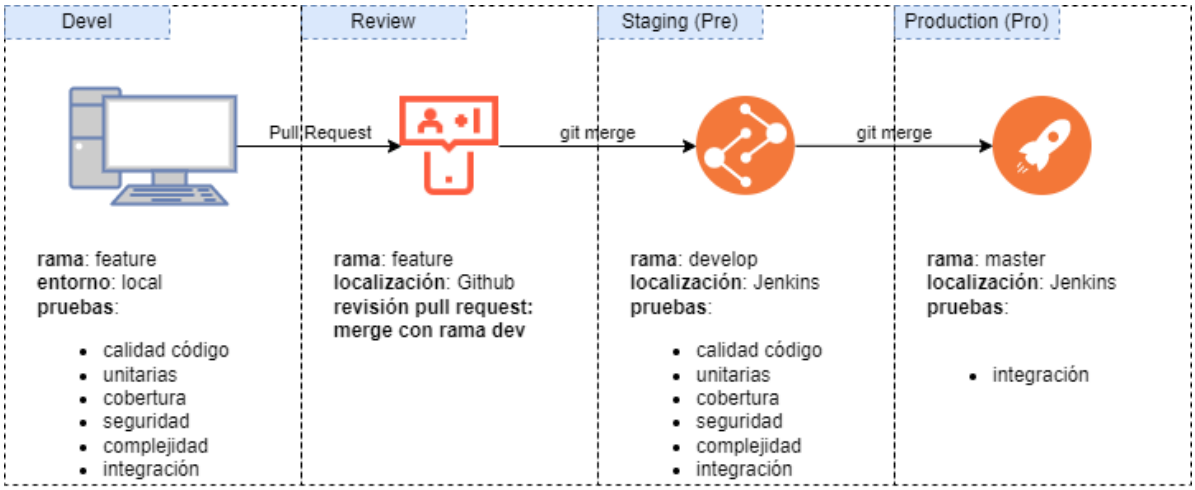
Resumen de la solución. Apartado A

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 con nombre de rama ***feature*.**
- 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**

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 mediante 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 Code*Pipeline*) 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.

Caso práctico 1. Apartado C (Optativo)

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, junto con las pruebas correspondientes, 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>

Material para entregar

Rellenar completamente el documento que lleva por nombre **Plantilla Solución CP1.docx**, con las secciones requeridas para los supuestos de los apartados A, B y C (de carácter optativo este último), 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 *pipeline*s de CI/CD para ambas aproximaciones o frameworks de operativización.

Nota: para su entrega, dicho documento de plantilla ha de exportarse 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.