Propuesta de solución

Caso práctico 1

URL de repositorio solución de GitHub:

Caso práctico 1. Apartado A

Una vez se ha determinado la correcta adecuación del aplicativo base dentro del ecosistema de servicios que provee Amazon Web Services a través del uso de [Serverless Framework](https://www.serverless.com/), es tiempo de dotar al caso práctico de una aproximación Full-AWS y Jenkins. ¿Qué significa? En línea con lo aprendido durante el programa, AWS dispone de una suite de servicios orientados a como disponibilizar 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.

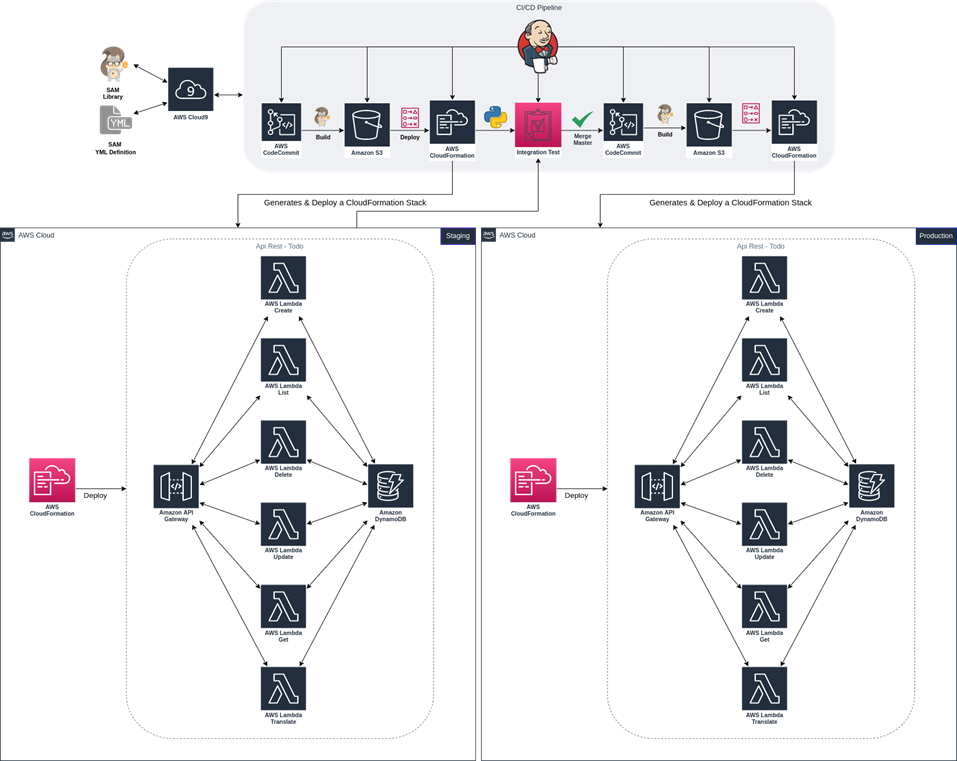
A continuación, se van a detallar brevemente cada uno de los servicios necesarios para la elaboración de este apartado:

* [AWS CloudFormation](https://aws.amazon.com/es/cloudformation/): Servicio de diseño, implementación y despliegue automático de infraestructura de aplicaciones Cloud, simplificando su diseño a través de un lenguaje común de modelado.
* [AWS CodeCommit](https://aws.amazon.com/es/codecommit/): Servicio autogestionado de control de código fuente, a través de un control del histórico de versiones en repositorios independientes.
* [Elastic Computer Service (EC2)](https://aws.amazon.com/es/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)](https://aws.amazon.com/es/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](https://www.jenkins.io/), desplegada dentro de una instancia EC2 para construir ahí los diferentes *pipelines* que se van a proponer en este apartado.

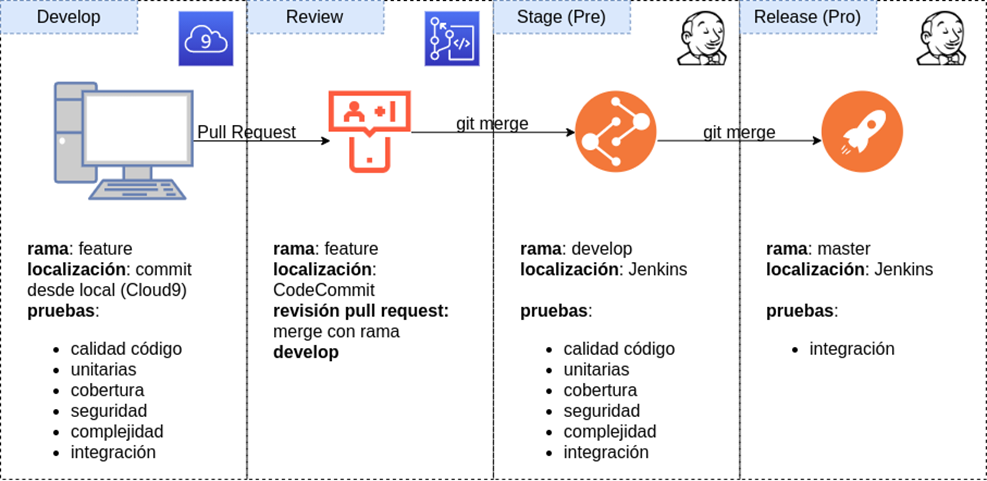
Resumen de la solución a implementar. Apartado A

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 ***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 B, 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**

|  |  |
| --- | --- |
| **Comando** | **Resultado** |
| $ git remote -v |  |

1. **Validación SAM CLI (Command Line Interface) y análisis de repositorio**
   1. **Ejecución del comando SAM funcionando:**

|  |  |
| --- | --- |
| **Comando** | **Resultado** |
| $ sam --version |  |

* 1. **Análisis del repositorio:** tras llevar a cabo la migración del repositorio entre sistemas de control de versiones de código, realizar la siguiente comprobación de la url de nueva creación de la ubicación destino:

|  |  |
| --- | --- |
| **Ficheros** | **Contenido** |
| **src** |  |
| **test** |  |
| **pipelines** |  |
| **template.yaml** |  |
| **samconfig.toml** |  |
| **localEnvironment.json** |  |

1. **Ejecución de proyecto en entorno local (SAM CLI):**
   1. **Pasos para levantar el entorno local:**

|  |  |
| --- | --- |
| **Resultados a mostrar** | **Salida** |
| **Crear red de docker** |  |
| **Levantar contenedor de docker** |  |
| **Crear tabla en dynamodb local** |  |
| **Empaquetar proyecto con SAM** |  |
| **Levantar la API localmente** |  |

Se adjunta el listado de peticiones a los métodos del servicio desplegados localmente con la url o *endpoint* relativa y un campo «resultado» al objeto de completitud por parte del alumno, así como los distintos pasos para conseguir desplegar **SAM** local, como las evidencias de que se ha realizado correctamente:

* 1. **En entorno local:**

|  |  |  |
| --- | --- | --- |
| **Función** | **In/out** | **Script** |
| **Create** | **comando** | curl -X POST http://127.0.0.1:8081/todos --data '{ "text": "Learn Serverless" }' |
| **resultado** |  |
| **List** | **comando** | curl http://127.0.0.1:8081/todos |
| **resultado** |  |
| **Get** | **comando** | curl http://127.0.0.1:8081/todos/<id> |
| **resultado** |  |
| **Update** | **comando** | curl -X PUT http://127.0.0.1:8081/todos/<id> --data '{ "text": "Learn Serverless", "checked": true }' |
| **resultado** |  |
| **Delete** | **comando** | curl -X DELETE http://127.0.0.1:8081/todos/<id> |
| **resultado** |  |

1. **Despliegue manual de aplicación SAM en Amazon Web Services:**

**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.

Completa 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** | **https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>** |
| **Update** | **https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>** |
| **Delete** | **https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id>** |

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

* 1. **Invocaciones a los métodos del API**

|  |  |  |
| --- | --- | --- |
| **Función** | **In/out** | **Script** |
| **Create** | comando | curl -X POST https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos --data '{ "text": "Learn Serverless" }' |
| resultado |  |
| **List** | comando | curl https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos |
| resultado |  |
| **Get** | comando | curl https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id> |
| resultado |  |
| **Update** | comando | curl -X PUT https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id> --data '{ "text": "Learn python and more", "checked": true }' |
| resultado |  |
| **Delete** | comando | curl -X DELETE https://XXXXXXX.execute-api.us-east-1.amazonaws.com/Prod/todos/<id> |
| resultado |  |

* 1. **Logs de SAM**

|  |  |
| --- | --- |
| **Resultados a mostrar** | **Salida** |
| **Log sam build** |  |
| **Log sam deploy** |  |

1. **Creación de pipelines de Jenkins para despliegue de arquitectura completa**
   1. **Pipeline de staging**

Al completar esta sección, 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 pipeline** |  |
| **Captura de pantalla del pipeline** |  |
| **Comentarios adicionales** |  |

* 1. **Pipeline de Production**

Indicar las salidas solicitadas del *pipeline*, bien copiando los logs, o adjuntando en la entrega final los ficheros de logs aparte.

|  |  |
| --- | --- |
| **Resultados a mostrar** | **Salida** |
| **Log pipeline** |  |
| **Captura de pantalla del pipeline** |  |
| **Explicación teórica de *rollback*** |  |
| **Comentarios adicionales** |  |

* 1. **/CD completo**

Indicar las salidas solicitadas del *pipeline*, bien copiando los logs, o adjuntando en la entrega final los ficheros de logs aparte.

|  |  |
| --- | --- |
| **Resultados a mostrar** | **Salida** |
| **Log pipeline** |  |
| **Captura de pantalla del pipeline** |  |
| **Comentarios adicionales** |  |

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 de 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.

|  |  |
| --- | --- |
| **Resultados a mostrar** | **Salida** |
| **Conclusiones finales** |  |
| **Pros** |  |
| **Contras** |  |
| **Arquitectura alternativa explicación** |  |
| **Arquitectura alternativa diagramas** |  |

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](https://docs.aws.amazon.com/comprehend/latest/dg/comprehend-dg.pdf) y [Translate de AWS](https://docs.aws.amazon.com/es_es/translate/latest/dg/translate-dg.pdf) si fueran necesarios. Una vez desarrollada la nueva función lambda, se ha de integrar con el resto de las 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> |  |
| /todos/<id>/en |  |
| /todos/<id>/fr |  |

Adicionalmente también se han de mostrar el código fuente desarrollado en la siguiente tabla, las inclusiones efectuadas en el fichero **template.yaml**, con el fin de tener la evidencia del trabajo realizado, así como las pruebas:

|  |  |
| --- | --- |
| **Pruebas** | **Detalle a entregar** |
| template.yaml |  |
| Bloque código pruebas unitarias |  |
| Bloque código pruebas integración |  |
| Bloque código pruebas calidad |  |
| Bloque código pruebas unitarias complejidad |  |

Pruebas

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

Material para entregar

Rellenar completamente el documento que lleva por nombre **plantilla\_solucion\_CP1.docx**, con las secciones requeridas para los supuestos de los apartados A y B, 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 se 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 (en la portada de este presente documento).**