



## ENTREGA 4 - ESCALABILIDAD EN LA CAPA BATCH/WORKER

### IMPLEMENTACIÓN SOLUCIONES ESCALABLES EN LA NUBE

#### Objetivos

- Conocer las consideraciones técnicas que deben ser tenidas en cuenta para escalar una aplicación web sobre la infraestructura de un proveedor IaaS público.
- Definir una estrategia de despliegue de código y de inicio de los servidores (web y workers) para que la aplicación pueda operar y escalar bajo demanda sin que el administrador intervenga.
- Implementar políticas de autoscaling que permitan que los servidores web de la aplicación puedan escalar automáticamente (autoscaling + monitorización).
- Implementar políticas de autoscaling que permitan que los procesos en batch (workers) puedan escalar automáticamente (autoscaling + monitorización).
- Utilizar un balanceador de carga que distribuya el volumen de peticiones entrantes entre los diferentes servidores web de la aplicación.
- Emplear un sistema escalable de paso de mensajes entre los servidores web y los workers, como es el caso de las colas.
- Ejecutar pruebas de estrés de la aplicación tanto para los servidores web como para los procesos que ejecutan trabajos en batch (workers).

#### Tiempo de dedicación

La presente entrega, correspondiente a la fase implantación en la nube, está programada para un periodo de **una semana**. Durante este tiempo, cada estudiante deberá destinar las horas asignadas en la planificación semanal. Se recuerda la importancia de conformar equipos de trabajo de acuerdo con las directrices establecidas en el curso, como condición clave para el éxito de la actividad.



## Lecturas previas

Material de lectura entregado durante el curso, en conjunto con la documentación ofrecida por AWS para los servicios de Amazon EC2, ELB, Amazon S3, Amazon RDS, Amazon SQS/SNS/Kinesis.

## Esquema de evaluación

La calificación de la entrega se distribuye de la siguiente manera:

- Actividades requeridas para la migración inicial de la aplicación: **70%**
- Documento de escenarios y resultados de las pruebas de estrés: **20%**
- Documento de la arquitectura de la aplicación: **10%**

## Recomendaciones y consideraciones

En esta entrega, se realizará el aprovisionamiento y despliegue en AWS de la aplicación web (API REST) desarrollada previamente. La aplicación deberá contar, como mínimo, con cuatro componentes esenciales para su ejecución en AWS: capa web, almacenamiento de objetos, la capa worker y la base de datos. Se podrá utilizar cualquier sistema de gestión de bases de datos que sea compatible con el servicio Amazon RDS.

En la fase de desarrollo, se sugiere emplear un contenedor docker con el mismo sistema de gestión de bases de datos que se utilizará en Amazon RDS. Cabe resaltar que en esta entrega se implementarán mecanismos de escalado automático (autoscaling) en la capa web y en la capa worker, y el sistema de almacenamiento de almacenamiento se mantendrá en buckets de Amazon S3.

Por otra parte, se recomienda activar las instancias en AWS únicamente durante la ejecución de las pruebas de capacidad y detenerlas cuando no sean necesarias, con el objetivo de optimizar el uso de los créditos asignados. Asimismo, al finalizar y cargar la entrega, se deberá eliminar la base de datos de Amazon RDS, dado que este servicio



genera un costo elevado. En caso de requerirse una sustentación síncrona, será obligatorio recrear la base de datos antes del encuentro con los tutores.

Dado que durante el proyecto se gestionarán credenciales de acceso a AWS, se aconseja no almacenarlas en texto plano dentro del código fuente de la aplicación, sino configurar variables de entorno para su manejo seguro. Para mayor protección y control de costos, se recomienda habilitar las alarmas de consumo y los presupuestos, con el fin de monitorear continuamente los gastos asociados a la cuenta de AWS.



## Restricciones de AWS ACADEMY

**Regiones soportadas:** Solo se puede trabajar en us-east-1 y us-west-2.

**Auto Scaling:** Amazon EC2 Auto Scaling puede asumir el rol IAM LabRole.

**AMIs soportadas:**

- Solo se aceptan AMIs en las regiones us-east-1 o us-west-2.
- Se permiten: AMIs de inicio rápido, personales (“Mis AMIs”) y de la comunidad.

**Tipos de instancias EC2 permitidos:** nano, micro, small, medium, large.

**Instancias bajo demanda únicamente:** No se permite usar instancias reservadas ni spot.

**Límites técnicos críticos:**

- **Máximo 9 instancias EC2 en ejecución simultánea por región**, sin importar el tipo.
- **Máximo 32 vCPUs en total** en ejecución simultánea.
- Intentar iniciar **20 o más instancias simultáneamente** desactiva la cuenta y elimina todos los recursos automáticamente.

**Restricciones en volúmenes EBS:**

- Solo hasta **100 GB**.
- Tipos permitidos: gp2, gp3, sc1 y estándar.

**Amazon Simple Notification Service (SNS)**

- Este servicio puede asumir el rol de IAM LabRole.

**Amazon Simple Queue Service (SQS)**

- Este servicio puede asumir el rol de IAM LabRole.

**Amazon Kinesis**



- Cuando desee crear un flujo de entrega de Kinesis, elija "Advance settings" (Configuración avanzada) y, luego, elija usar el **LabRole** existente.

#### **Gestión de sesiones del laboratorio:**

- Las instancias activas se detienen al finalizar una sesión.
- Si estaban protegidas contra detención, dicha protección se elimina.
- Al iniciar una nueva sesión, las instancias detenidas se reactivan con una **nueva IP pública**, salvo que se use una IP elástica.

#### **Recomendaciones clave:**

- Detenga las instancias cuando termine el trabajo diario.
- Revise las instancias activas al iniciar una nueva sesión para evitar consumo no deseado del presupuesto.



## Modelo de Despliegue - Escalabilidad en la Capa Batch o Worker

Con el fin de lograr que la aplicación web esté diseñada para escalar, se deberá modificar el aplicativo desarrollado en la entrega 3 tal que sea capaz de responder a la demanda de usuarios requerida. La organización ya ha identificado que la aplicación deberá implementar los siguientes servicios:

- **Amazon EC2:** Ejecución de la capa web y la capa worker. Por decisión de negocio, se han seleccionado instancias de cómputo con 2 vCPU, 2 GiB de RAM y 30 GiB de almacenamiento.
- **Amazon RDS:** Almacenamiento de la información de la aplicación en una base de datos relacional. En las fases iniciales de la implementación, puede utilizarse una instancia de EC2 y sustituirla por RDS únicamente durante las pruebas de carga.
- **Amazon S3:** Almacenamiento de los videos originales y los procesados.
- **Amazon CloudWatch:** Monitoreo de las instancias y los demás servicios empleados.
- **Autoscaling:** Servicio para escalar la capa web en función de los usuarios concurrentes.
- **Load Balancer:** Distribución de la carga entre las instancias que exponen el API REST.
- **Amazon SQS o Amazon Kinesis:** Intercambio de mensajes de forma confiable, rápida y asíncrona.

Esta entrega es una extensión del modelo despliegue de la entrega anterior, es decir, abarca las características y requerimientos previos, además de los ajustes solicitados a continuación. En concreto, el despliegue incluye el uso de **un balanceador de carga, hasta 3 servidores web, hasta 3 servidores workers, políticas de autoscaling y**



**sistema de mensajes.** A fin de implementar el modelo propuesto, es necesario llevar a cabo las siguientes actividades:

1. Configurar el servicio de balanceo de carga para poder desplegar varios servidores web.  
**Actividad completada en la entrega anterior.**
2. Definir e implementar una estrategia para que los servidores web escalen de manera automática. Se deberán establecer las condiciones de escalamiento mediante los servicios de monitoreo, autoscaling y de balanceo de cargas. **Actividad completada en la entrega anterior.**
3. **(20%)** Diseñar e implementar una estrategia para que los servidores de procesamiento de archivos (workers) puedan escalar de manera automática. Se deberán establecer las condiciones de escalamiento utilizando los servicios de monitoreo, autoscaling y cola de mensajes.
4. **(20%)** Configurar el servicio de mensajería asíncrona (SQS o Kinesis), el cual actuará como sistema de comunicación entre los servidores web y los procesos workers. Se deberán efectuar las modificaciones necesarias para que ahora los servidores web coloquen las solicitudes para procesar nuevos videos en el servicio de mensajería asíncrona (SQS o Kinesis), y los workers extraigan los archivos de ella.
5. **(20%)** Configurar la capa web (API REST) de la aplicación en alta disponibilidad. Para esto configure el balanceador de carga y los nodos tal que se desplieguen sobre dos zonas de disponibilidad de AWS.
6. **(10%)** La aplicación web debe satisfacer la totalidad de los requerimientos funcionales que ya fueron estipulados y detallados en el enunciado del proyecto.

## Documentación: Arquitectura de la aplicación

Se deberá entregar un documento que describa la arquitectura ajustada, incluyendo una breve explicación de las tecnologías y servicios incorporados, así como una indicación clara de los cambios realizados con respecto a la entrega anterior. Es fundamental presentar tanto el modelo de despliegue como el modelo de componentes; no obstante, se pueden incluir otros recursos visuales que se consideren pertinentes para complementar la documentación.





Toda la documentación deberá estar alojada en el repositorio de GitLab, dentro del directorio dedicado (**/docs/entrega4**), y deberá estar referenciada en el archivo **README.md** para facilitar su acceso. Se recomienda utilizar esta estructura para las entregas posteriores.

## Documentación: Análisis de capacidad

Lea el documento “Entrega - Análisis de capacidad” para conocer toda la especificación del entregable solicitado. Los porcentajes de esta actividad se dividen en los siguientes ítems de calificación:

- **(10%)** Pruebas de estrés, análisis y documentación - Escenario 1.
- **(10%)** Pruebas de estrés, análisis y documentación - Escenario 2.

Además, incorpore las conclusiones derivadas de las pruebas de estrés ejecutadas y las consideraciones adicionales para continuar escalando la aplicación web y así atender a los cientos de usuarios finales que van a subir archivos o consumir recursos del API REST de manera concurrente. En otras palabras, incluya las modificaciones, en cuanto a las características de los recursos empleados, por ejemplo, para que la siguiente versión del proyecto pueda admitir más clientes y tareas procesadas.

El **de análisis de capacidad** debe ser organizado y entregado dentro del repositorio del proyecto, dentro de la carpeta **/capacity-planning**, donde se almacenará el documento correspondiente.

El reporte debe almacenarse en un archivo llamado **pruebas\_de\_carga\_entrega4.md**, el cual debe incluir el análisis detallado de capacidad de la aplicación, los resultados de los escenarios de carga planteados y las recomendaciones para escalar la solución. Esta estructura debe mantenerse de forma consistente en las futuras entregas del proyecto.

## Entregables

1. Aplicación desplegada y en ejecución sobre AWS.
2. Documento con la descripción de la arquitectura de solución planteada.





3. Documento con el plan de pruebas de carga refinado.
4. Documento de análisis de las pruebas de carga.

Por lo cual, se solicita:

- a. Crear un release del código fuente en el repositorio del grupo en GitLab.
- b. Entregar toda la documentación vía GitLab.

Adicionalmente, cada grupo debe preparar un video de sustentación de máximo 20 minutos, donde explore la solución propuesta y efectúe pruebas sobre cada uno de los endpoints, empleando distintos parámetros de entrada. En especial, emplee vídeos con características (duración y relación de aspecto) variadas que permitan evidenciar en profundidad el comportamiento de la aplicación. A petición de los tutores del curso, podría programarse una sustentación síncrona, para la cual es indispensable que la aplicación se encuentre desplegada y en ejecución. Además, debe tener configurado el acceso a la infraestructura y los recursos necesarios para poder realizar una prueba de desempeño si así lo solicita el tutor.