## **Plan para alojar la aplicación web en AWS Lab y ponerla en producción**

### **Distinción entre IaaS, PaaS y SaaS**

1. **IaaS (Infraestructura como Servicio):**Este modelo ofrece recursos básicos como servidores virtuales, almacenamiento y redes, dejando al usuario la responsabilidad de configurarlos y gestionarlos. Por ejemplo, Amazon EC2 permite crear y administrar instancias virtuales.
2. **PaaS (Plataforma como Servicio):**Proporciona una plataforma preconfigurada para desarrollar y desplegar aplicaciones sin preocuparte por la infraestructura subyacente. Un ejemplo es AWS Elastic Beanstalk, que automatiza tareas como la provisión de servidores y configuración de entornos.
3. **SaaS (Software como Servicio):**Consiste en aplicaciones listas para ser usadas directamente por los usuarios. En AWS, un ejemplo es Amazon WorkSpaces para escritorios virtuales. Otros ejemplos fuera de AWS incluyen servicios como Google Drive o Gmail.

### **Tipos de instancias EC2, diferencias y tarifas**

**AWS EC2** es un servicio IaaS que permite crear servidores virtuales. Los principales tipos de instancias son:

* **T (Uso General):** Económicas y diseñadas para cargas ligeras o intermitentes, como desarrollo o pruebas. Ejemplo: t2.micro (gratis en la capa gratuita).
* **M (Uso General Equilibrado):** Ofrecen un balance entre CPU, memoria y red, ideales para aplicaciones web o bases de datos pequeñas. Ejemplo: m5.large.
* **C (Optimizadas para Computación):** Adecuadas para tareas intensivas de CPU, como cálculos matemáticos o simulaciones. Ejemplo: c5.xlarge.
* **R (Optimizadas para Memoria):** Diseñadas para aplicaciones con alta demanda de RAM, como análisis de datos. Ejemplo: r5.large.
* **G (Optimizadas para GPU):** Ideales para gráficos o aprendizaje automático. Ejemplo: g4dn.xlarge.

**Tarifas:**Varían según el tipo de instancia y la región. Por ejemplo, t2.micro es gratuita durante el primer año en la capa gratuita, mientras que instancias avanzadas como c5.xlarge cuestan entre $0.17 y $0.68 por hora.

### **Requisitos para acceder y permisos necesarios**

* **Acceso a la cuenta:** Necesitas registrarte con un correo electrónico y un método de pago válido.
* **Permisos requeridos:**
  + ec2:CreateInstance para lanzar instancias.
  + ec2:DescribeInstances para listar instancias existentes.
  + ec2:TerminateInstance para detenerlas.
* **Credenciales:** Es indispensable contar con la *Access Key ID* y el *Secret Access Key* para interactuar con AWS desde la línea de comandos o aplicaciones externas.

### **Conexión y colaboración con instancias**

* **Conexión:**Puedes conectarte mediante:
  + Navegador web (AWS Console): Abre una terminal en la máquina desde el navegador.
  + **SSH:** Utilizando una clave privada generada al crear la instancia.
* **Colaboración:**Puedes compartir la instancia con tus compañeros proporcionando las claves pública y privada o creando cuentas de usuario dentro de la instancia con permisos específicos.

### **Despliegue de la aplicación Django con Docker y HTTPS**

Para desplegar tu aplicación Django junto con las de tus compañeros:

1. **Preparación del proyecto:**
   * Crea un archivo Dockerfile para definir cómo se construirá tu aplicación.
   * Genera un archivo docker-compose.yml para gestionar servicios como la aplicación y la base de datos.
2. **Gestión del código:**
   * Usa **AWS CodeCommit** para alojar y gestionar el repositorio del proyecto.
3. **Configuración de dominio y HTTPS:**
   * Registra un dominio en DuckDNS o Route 53.
   * Configura un certificado HTTPS con Let's Encrypt.
4. **Despliegue:**
   * Ejecuta los contenedores Docker en una instancia EC2.
   * Configura un servidor web (como Nginx) para servir la aplicación.

### **Pruebas de velocidad de la aplicación**

Utiliza herramientas como **AWS CloudWatch** para monitorizar métricas como la carga de la instancia, latencia y tiempos de respuesta.

### **Creación de una instancia RDS gratuita de PostgreSQL**

1. **Configuración:**
   * Desde la consola, selecciona PostgreSQL y configúralo como público o con acceso restringido a tu instancia EC2.

**Integración con Django:**En el archivo settings.py incluye:  
python  
Copiar código  
DATABASES = {

'default': {

'ENGINE': 'django.db.backends.postgresql',

'NAME': 'nombre\_bd',

'USER': 'usuario',

'PASSWORD': 'contraseña',

'HOST': 'endpoint-rds.amazonaws.com',

'PORT': '5432',

}

}

1. **Permisos necesarios:**
   * rds:CreateDBInstance para crear instancias.
   * rds:Connect para acceder a la base de datos.

**Ventajas de RDS:**Gestión automática de copias de seguridad, alta disponibilidad y escalabilidad.  
**Desventajas:**Mayor costo en comparación con instalar PostgreSQL directamente en EC2.

### **Uso de AWS Elastic Beanstalk**

**AWS Elastic Beanstalk** es un servicio PaaS que facilita el despliegue de aplicaciones. Basta con subir tu código y el servicio se encarga de configurar la infraestructura: servidores EC2, bases de datos, balanceadores, etc.

* **Permisos necesarios:**elasticbeanstalk:CreateEnvironment para crear entornos.
* **Costos:**No tiene costo adicional, pero se cobran los recursos subyacentes como EC2 y RDS.
* **Ventajas:**Fácil de usar y con gestión automática de la infraestructura.
* **Desventajas:**Ofrece menos control y puede ser más costoso.

En resumen, AWS ofrece herramientas flexibles para desplegar aplicaciones, desde un control granular con EC2 hasta opciones automatizadas con Elastic Beanstalk.