Proyecto Opcional

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Redes (IC 7602)
Primer Semestre 2025

1. **Objetivo General**
   - ➔ Desarrollar habilidades en tecnologías, servicios y lenguajes de programación que serán utilizados durante el curso.
2. **Objetivos Específicos**
   - ➔ Instalar servicios de red.
   - ➔ Configurar acceso a servicios de red mediante Kubernetes..
   - ➔ Automatizar el aprovisionamiento de infraestructura mediante Helm y Kubectl.
   - ➔ Automatizar la instalación de servicios de red mediante Docker/Kubernetes.
3. **Datos Generales**
   - ➔ El valor del proyecto: 15%
   - ➔ La tarea debe ser implementada en grupos de máximo 5 personas.
   - ➔ **El proyecto es opcional, el porcentaje obtenido será sumado a la nota final del curso**.
   - ➔ La **fecha de entrega** es 14 de marzo del 2025 antes de las 11:59 pm, la revisión será realizada el día 16 de marzo del 2025, las citas de revisión serán enviadas por el profesor.
   - ➔ Cualquier indicio de copia será calificado con una nota de 0 y será procesado de acuerdo con el reglamento. La copia incluye código/configuraciones que se puede encontrar en Internet y que sea utilizado parcial o totalmente sin el debido reconocimiento al autor.
   - ➔ La revisión es realizada de forma virtual mediante citas en las cuáles todas las personas integrantes del grupo deben estar presentes, el proyecto debe estar completamente automatizado, el profesor decide en que computadora probar.
   - ➔ Se debe incluir documentación con instrucciones claras para ejecutar el proyecto.
   - ➔ Se deben incluir pruebas unitarias en caso de ser necesarias, esto con el fin de verificar los componentes individuales de su proyecto, si estas no están presentes se obtendrá una nota de 0.
   - ➔ Se espera que todas las personas integrantes del grupo entiendan la implementación suministrada.
   - ➔ Se deben seguir buenas prácticas de programación en el caso de que apliquen, entre ellas:
     - ◆ Documentación interna y externa.
     - ◆ Estándares de código.
     - ◆ Diagramas de arquitectura.
     - ◆ Diagramas de flujo
     - ◆ Pruebas unitarias.
   - ➔ Toda documentación debe ser implementada en Markdown.
   - ➔ Si el proyecto no se encuentra automatizado obtendrá una nota de 0. Si el proyecto no se entrega completo, la porción que sea entregada debe estar completamente automatizada, de

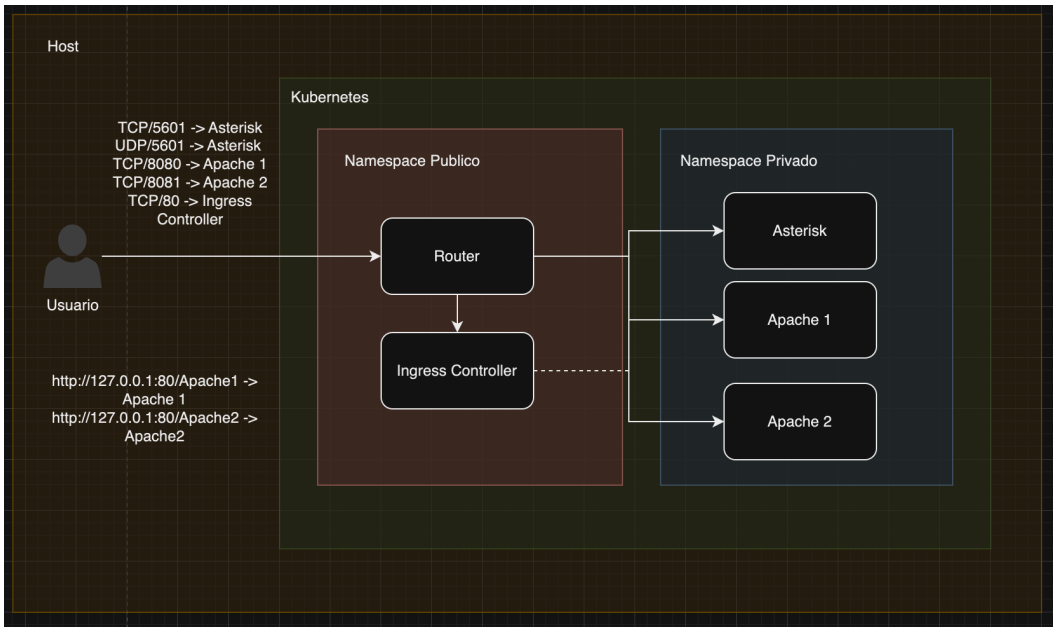
lo contrario se obtendrá una nota de 0.

- ➔ En caso de no entregar documentación se obtendrá una nota de 0.
- ➔ El email de entrega debe contener una copia del proyecto en formato tar.gz y un enlace al repositorio donde se encuentra almacenado, debe seguir los lineamientos en el programa de curso.
- ➔ Los grupos de trabajo se deben autogestionar, la persona facilitadora del curso no es responsable si un miembro del equipo no realiza su trabajo.
- ➔ En consulta o en mensajes de correo electrónico o Telegram, como mínimo se debe haber leído del tema y haber tratado de entender sobre el mismo, las consultas deben ser concretas y se debe mostrar que se intentó resolver el problema en cuestión.

4. **Descripción**

El presente proyecto busca familiarizar a las personas estudiantes con servicios de capa de aplicación, capa de red y capa de transporte que serán estudiados durante el curso, para lograr esto se utilizarán herramientas/tecnologías de automatización de infraestructura y de configuración, es importante mencionar que todas estas tecnologías de automatización son utilizadas extensivamente por muchas organizaciones como parte del procesos de DevOps, CI/CD y arquitectura en Cloud/Microservicios, las mismas son un insumo muy importante para las entrevistas de practica de especialidad.

El siguiente diagrama muestra la arquitectura requerida:



- Namespaces: Se deberán crear dos Namespaces con nombres privado y público, los Namespaces son contenedores de recursos en Kubernetes (así como en tecnologías de Cloud

como Azure Container App, AWS ECS y GCP Cloud Run), pero estos tienen un impacto en la forma en la cual se acceden a recursos mediante servicios de red como DNS, que es un servicio que se estudiará a profundidad al final del curso.

- Dentro del namespace público, se deberán crear los siguientes recursos como Deployments con una réplica y un Service de tipo NodePort:
    - Router: Se recomienda utilizar una imagen base de Ubuntu e implementarlo con IPtables, aunque los grupos pueden decidir utilizar la combinación de tecnologías y sistemas operativos de su preferencia, el router expondrá los puertos:
        - TCP/5601: El tráfico entrante a este puerto se debe redireccionar hacia Asterisk.
        - UDP/5601: El tráfico entrante a este puerto se debe redireccionar hacia Asterisk.
        - TCP/8080: El tráfico entrante a este puerto se debe redireccionar hacia Apache 1.
        - TCP/8081: El tráfico entrante a este puerto se debe redireccionar hacia Apache 2.
        - TCP/80: El tráfico entrante a este puerto se debe redireccionar hacia el Ingress Controller.
    - Ingress Controller: Este es un componente de capa de aplicación que permite manipular peticiones HTTP y redireccionar las mismas basado en características de un URI, existen múltiples implementaciones, pero por facilidad se invita a los grupos a utilizar una implementación basada en Nginx (aunque pueden utilizar cualquier otra), el mismo tendrá la siguiente funcionalidad:
        - Cuando un cliente ingrese a http://127.0.0.1:80/Apache1 el request será redireccionado al Apache1.
        - Cuando un cliente ingrese a http://127.0.0.1:80/Apache2 el request será redireccionado al Apache2.
- Dentro del namespace privado, se deberán crear los siguientes recursos como Deployments con una réplica y un Service de tipo ClusterIP:
    - Apache1 y Apache2: Son simples websites que al ingresar a los mismos, mostraran “Hola Apache1” y “Hola Apache 2”
    - Asterisk (ver sección).

Para implementar estos servicios, se debe utilizar la herramienta Helm, que permite generar infraestructura Kubernetes como código, esta herramienta permite trabajar con los mayores Cloud Providers así como de forma local.

FreePBX / Asterisk

Asterisk y FreePBX se utilizan en conjunto para desarrollar centrales telefónicas, el mismo utiliza el protocolo de capa de sesión llamado SIP, este protocolo hace uso de los protocolos de capa de transporte UDP y TCP, específicamente utiliza el puerto 5060.

Existen múltiples guías en Internet de como instalar estas dos tecnologías y en las páginas web de cada tecnología encontraran paso a paso de la forma de realizar la instalación, aquí pueden encontrar una guía útil para realizar la instalación de ambas en Ubuntu.
Se deben crear extensiones SIP y hacer llamadas mediante algún cliente de SIP, existen clientes para IPhone, Android y desktops.

Pruebas

- Las pruebas de servicios HTTP como lo son los Apache servers y el Ingress Controller, se puede realizar fácilmente con cualquier internet browser como Chrome, también es posible utilizar herramientas como curl y wget.
- Se pueden realizar pruebas de conexión a nivel de capa de red y de transporte mediante la herramienta telnet.
- Mediante la herramienta Kubectl específicamente el comando port-forward es posible acceder directamente a servicios de red.
- El funcionamiento de Asterisk puede ser verificado con algún cliente de protocolo SIP, uno de los más utilizados es Zoiper.

Documentación

Se deberá entregar una documentación que al menos debe incluir:

- Instrucciones claras de como ejecutar su proyecto.
- Pruebas realizadas, con pasos para reproducirlas.
- Resultados de las pruebas unitarias.
- Recomendaciones y conclusiones (al menos 10 de cada una).
- La documentación debe cubrir todos los componentes implementados o instalados/configurados, en caso de que algún componente no se encuentre implementado, no se podrá documentar y tendrá un impacto en la completitud de la documentación.
- Referencias bibliográficas donde aplique.

Imaginen que la documentación está siendo creada para una persona con conocimientos básicos en el área de computación y ustedes esperan que esta persona pueda ejecutar su proyecto y probarlo sin ningún problema, el uso de imágenes, diagramas, code snippets, videos, etc. son recursos que pueden ser útiles.
La documentación debe estar implementada en Markdown y compilada en un PDF.

5. **Recomendaciones**
    - Utilizar telnet/curl/postman para interactuar con los componentes y verificar que los mismos están funcionando correctamente.

- Utilizar Docker Desktop para realizar la instalación de Kubernetes.
- Realicen peer-review/checkpoints semanales y no esperen hasta el último momento para integrar su aplicación.
- Crear un repositorio de GitHub e invitar al profesor.
- Realizar commits y push regulares.
- Utilizar mobaxterm en Windows para acceder fácilmente a pods corriendo en Kubernetes.

6. **Entregables**

- Documentación.
- Archivos de automatización.

7. **Evaluación**

Funcionalidad / Requerimiento	Porcentaje
Documentación	20%
Implementación (*): ● Archivos de automatización Kubernetes (20%) ● Apache Servers (10%) ● Router (10%) ● Ingress Controller (15%) ● FreePBX / Asterisk (25%)	80%
	100%

(*) Todo tiene que estar debidamente automatizado, de lo contrario se calificará con una nota de 0.