Proyecto Opcional

Tecnológico de Costa Rica
Escuela de Ingeniería en Computación
Redes (IC 7602)
Segundo Semestre 2022

1. **Objetivo General**
   - ➔ Desarrollar habilidades en tecnologías y lenguajes de programación que serán utilizados durante el curso.
2. **Objetivos Específicos**
   - ➔ Desarrollar arquitecturas cliente/servidor UDP y TCP en lenguajes de programación C y Python.
   - ➔ Automatizar una solución mediante el uso de Docker, Docker Compose, Kubernetes y Helm Charts.
   - ➔ Desarrollar arquitecturas de microservicios en Kubernetes.
   - ➔ Implementar un sitio web y un REST API en Python.
   - ➔ Instalar y configurar servicios de bases de datos NoSQL.
3. **Datos Generales**
   - ➔ El valor del proyecto: 15%
   - ➔ Nombre del proyecto: CrazyChess.
   - ➔ La tarea debe ser implementada en grupos de máximo 4 personas.
   - ➔ **El proyecto es opcional, el porcentaje obtenido será sumado a la nota final del curso**.
   - ➔ La **fecha de entrega** es 26 de agosto del 2022 antes de las 8:53 pm.
   - ➔ 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 todos los y las 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 para verificar los componentes individuales de su proyecto, si estas no están presentes se obtendrá una nota de 0.
   - ➔ Se espera que todos y todas las 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**

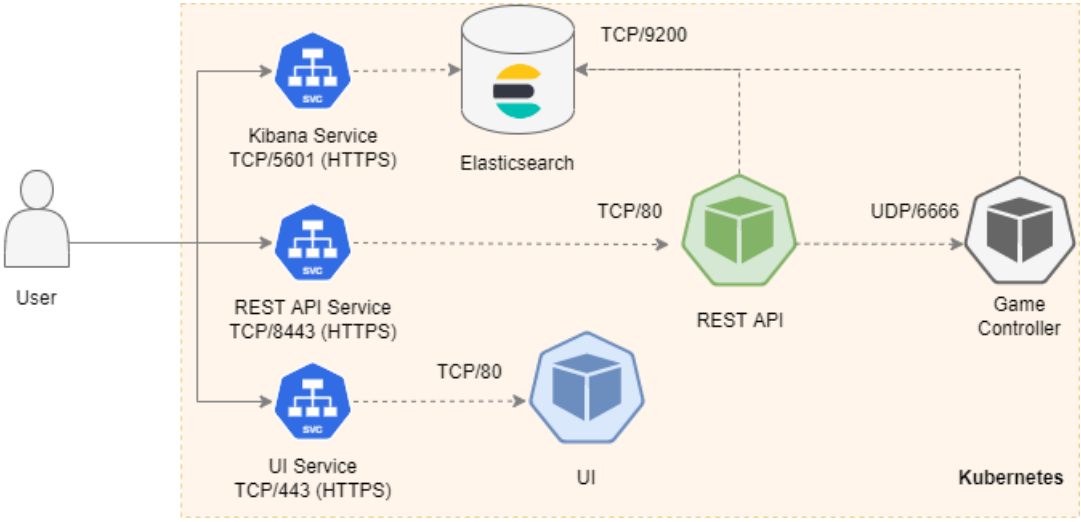
El proyecto opcional buscar dar a todos y todas las estudiantes la oportunidad de desarrollar habilidades en las tecnologías y lenguajes de programación que serán utilizados durante el curso para tareas cortas y proyectos.

El presente proyecto busca desarrollar un juego de ajedrez llamado, *CrazyChess*, el mismo utilizara extensivamente clientes y servidores TCP/UDP los cuales pertenecen a la capa de transporte que será estudiada en detalle durante el curso, así como las tecnologías que están dominando el mercado como lo son Kubernetes y bases de datos NoSQL como Elasticsearch, todo esto soportado por herramientas de automatización como lo son Docker, Docker Compose y Helm Charts.

El funcionamiento del juego es simple, múltiples usuarios se pueden conectar a una interfaz gráfica de usuario (UI), el mismo puede decidir entre tres opciones:

- Anfitrión de partida: En esta opción, el usuario selecciona si quiere seguir con una partida para este caso se le solicita el numero de partida o si quiere iniciar una partida, este caso se le retorna un numero aleatorio de partida, seguidamente el usuario ingresa a un tablero de ajedrez donde puede jugar si es su turno.
- Invitado de partida: Al ingresar como invitado, el usuario se le presenta una partida aleatoria previamente iniciada por un anfitrión, luego de hacer un movimiento, se le presenta otra partida aleatoria o se pone en espera hasta que una partida necesite un movimiento, la partida que se presenta no es exclusiva, esto quiere decir por ejemplo, que si existe una partida abierta por un anfitrión y existen 20 invitados, estos tendrán la oportunidad de realizar el movimiento y funcionara de la siguiente forma, en el momento que se recibe el primer movimiento, se espera 20 segundos para recibir mas movimientos por parte de otros usuarios, el movimiento que mas se repite es el escogido, en caso de empate, se escoge un movimiento aleatorio.
- Anfitrión e Invitado de partido: Juega como anfitrión e invitado simultáneamente.

El siguiente diagrama presenta los componentes que deberán ser implementados:



Game Controller (GC): Este componente se implementará como uno o más pods en lenguaje de programación C, que escuchan en el puerto UDP/6666, los mismos deberán vivir en un Deployment que permitirá configurar las réplicas deseadas, estas replicas deberán encontrarse detrás de un servicio de tipo ClusterIP.
Este componente por medio del servicio de tipo ClusterIP, será accedido por el componente REST API, toda comunicación entre estos dos componentes, deben intercambiar mensajes JSON codificados en BASE64.
GC realiza las siguientes funciones:

- Crear juegos.
- Recibir movimientos.
- Aplicar movimientos (luego de pasados los 20 segundos)
- Toda la información del juego se mantendrá en un índice de Elasticsearch.
- Deberá guardar un histórico de toda la actividad realizada en un índice de Elasticsearch.
- Debe validar que los movimientos válidos.

REST API: Este componente se implementará como uno o más pods en lenguaje de programación Python, el mismo implementa un RESTful API mediante el cual se expone toda la funcionalidad requerida para poder jugar ajedrez de la forma explicada anteriormente, para exponer información se usarán métodos GET y para modificar/crear se usarán PUT/POST, este componente escucha en el puerto TCP/80, los mismos deberán vivir en un Deployment que permitirá configurar las réplicas deseadas, estas replicas deberán encontrarse detrás de un servicio de tipo ClusterIP. Las funciones creadas deberán estar debidamente documentados utilizando Swagger y se recomienda utilizar el módulo de Python llamado Flask.
Como muestra el diagrama, este componente habla tanto con Elasticsearch, así como GC, en el caso del primero es únicamente para obtener información y guardar un request_log, en el caso del segundo interactúa para todo lo relacionado con el juego de ajedrez.

El request_log se guarda como un índice en Elasticsearch y captura todos los requests realizados al REST API.

UI: Se debe implementar un UI para jugar ajedrez, la misma interactúa con el REST API, se puede implementar en cualquier tecnología, el único requerimiento es que este completamente automatizado y que se ejecute como uno o más pods en Kubernetes.

Elasticsearch: Existen múltiples implementaciones de Kubernetes/Helm para instalar Elasticsearch y su interfaz grafica Kibana, para mencionar algunas:

- Elastic Cloud on Kubernetes
- Elasticsearch Helm Charts
- Bitnami Elasticsearch Stack

Las mismas instalan completamente Elasticsearch/Kibana.

Services: Los componentes Elasticsearch/Kibana, UI y REST API se deben exponer hacia la maquina host (donde se ejecuta Kubernetes) utilizando servicios de tipo NodePort, deben utilizar HTTPs.

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 instalar Kubernetes en Windows.
- Utilizar Minikube para ejecutar Kubernetes en Linux
- 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.

6. **Entregables**

- Documentación.
- Docker files, Docker Compose files y Helm Charts junto con todos los archivos/scripts requeridos para ejecutar su proyecto.

7. **Evaluación**

Funcionalidad / Requerimiento	Porcentaje
Documentación	10%
Implementación (*): ● Game Controller (30%) ● REST API (30%) ● UI (20%) ● Elasticsearch/Kibana (5) ● Servicios (5)	90%
	100%

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