Arquitecturas distribuidas Big Data

Sistema de Rescate y Emergencias Espaciales

Por: Brianny Hernández

Fase 1: Definición del tema y selección de herramientas

El Sistema de Rescate y Emergencias Espaciales es una arquitectura distribuida diseñada para proporcionar monitoreo en tiempo real, análisis de datos y respuesta rápida ante situaciones de emergencia en el espacio. Este sistema integra diversas fuentes de datos provenientes de naves espaciales, estaciones espaciales, satélites y otros activos espaciales para detectar, analizar y responder a eventos críticos, garantizando la seguridad de las misiones y los tripulantes.

Esta imagen fue creada por Leonardo.Ai

Dentro de las funcionalidades de este sistema, se pueden mencionar:

1. Reporte de emergencias: Los tripulantes contaran con la aplicación y podrán reportar que se encuentran en una emergencia mediante un botón que activara una alarma en el sistema.
2. Monitoreo continuo: Sensores instalados en naves espaciales, estaciones y satélites para recoger datos en tiempo real.
3. Detección de Anomalías: Notificaciones automáticas cuando se detecten condiciones fuera de los parámetros normales.
4. Notificación de emergencias: Por otro lado, los rescatistas que se encuentren más cerca del incidente serán notificados una vez se reporte una emergencia para que provean asistencia.

5. Evaluación de Impacto: Análisis del impacto potencial de una emergencia detectada, incluyendo el riesgo para los tripulantes y la misión.
6. Coordinación de Rescate: Planificación y ejecución de maniobras de rescate, incluyendo la redirección de naves y el despliegue de cápsulas de escape.

Todas estas funcionalidades se corresponden con los objetivos de este sistema de rescate:

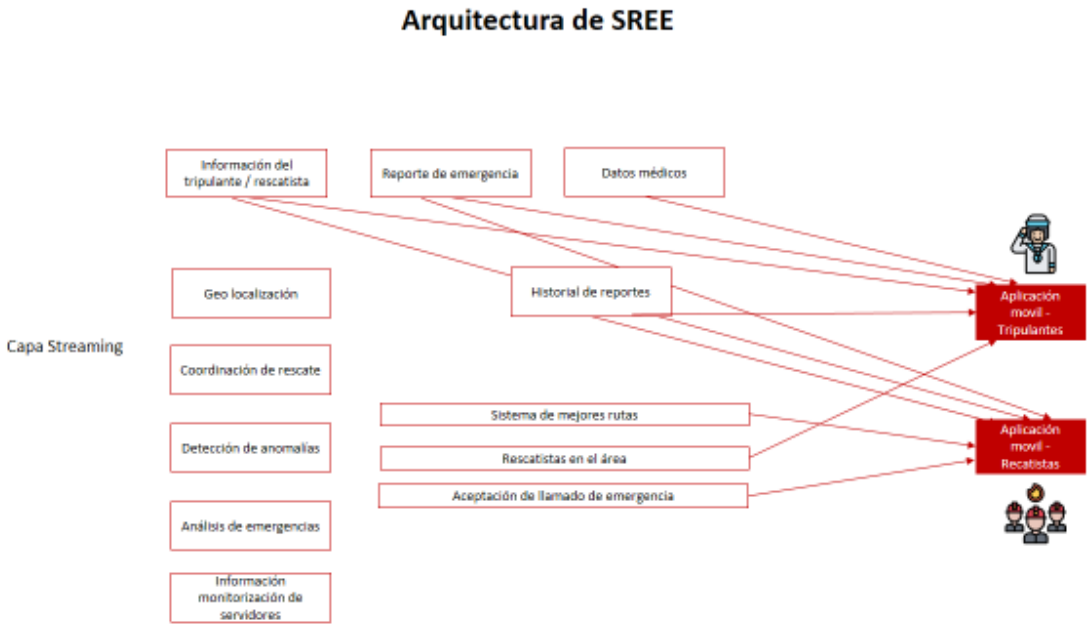
- Implementar un sistema de monitoreo continuo de todos los activos espaciales para detectar situaciones de emergencia, como fallos en los sistemas, colisiones inminentes o condiciones ambientales adversas.
- Utilizar análisis avanzados y técnicas de machine learning para interpretar grandes volúmenes de datos espaciales, identificar patrones de riesgo y prever posibles emergencias.
- Desarrollar protocolos y sistemas automáticos para responder rápidamente a situaciones de emergencia, incluyendo la activación de procedimientos de rescate y la comunicación con las agencias de rescate.
- Facilitar la comunicación y coordinación entre diferentes agencias espaciales y equipos de rescate para asegurar una respuesta integrada y efectiva.
- Garantizar una gestión eficiente de los recursos disponibles para el rescate y la gestión de emergencias, optimizando el uso de energía, combustible y otros suministros críticos.

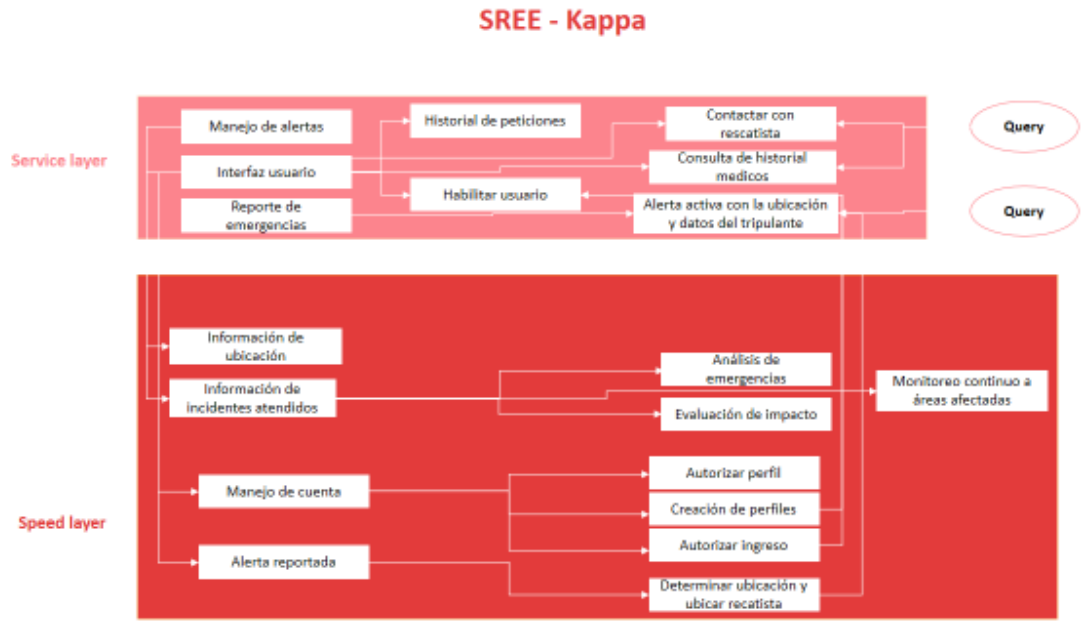
Herramientas

Para diseñar y desarrollar un Sistema de Rescate y Emergencias Espaciales (SREE) eficiente y robusto, se requiere una arquitectura distribuida que utilice tecnologías y herramientas avanzadas capaces de manejar grandes volúmenes de datos y proporcionar análisis y respuestas en tiempo real. A continuación se presentan las principales herramientas y tecnologías recomendadas para este proyecto.

- Hadoop Distributed File System (HDFS): permite almacenar grandes volúmenes de datos espaciales de manera eficiente y segura, asegurando alta disponibilidad y tolerancia a fallos.
- Apache YARN (Yet Another Resource Negotiator): permite la gestión eficiente de recursos y la ejecución paralela de tareas, optimizando el uso del clúster.
- Apache Hive: facilita el análisis de datos almacenados en HDFS, proporcionando una capa de abstracción SQL sobre los datos distribuidos.

Fase 2: Diseño y desarrollo de la arquitectura



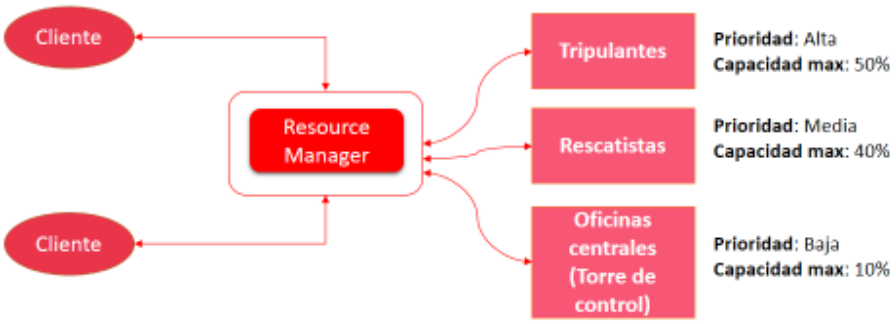


Las emergencias espaciales requieren una respuesta inmediata para prevenir desastres y salvar vidas y debido a que la arquitectura Kappa está diseñada para el procesamiento en tiempo real, lo que permite detectar y responder a eventos tan pronto como ocurren. A diferencia de Lambda,

que requiere mantener dos caminos de procesamiento (batch y stream), Kappa utiliza un único camino de procesamiento de datos, lo que simplifica el diseño y la implementación.

Esta arquitectura garantiza que siempre se procesan los datos más recientes, lo cual es crucial para la detección y respuesta rápida a emergencias ya que está optimizada para minimizar la latencia, procesando los datos tan pronto como llegan.

Diagrama YARN de SREE



Consideraciones para el diagrama YARN

Tripulantes: Estos deben tener la mayor prioridad y por ende capacidad mayor que el resto debido a que son los que reportan los incidentes y llenan los detalles del reporte a realizar.

Rescatistas: Su prioridad es media pero aún tienen una capacidad muy parecida a los tripulantes ya que deben responder a las alertas en cuanto aparezcan y visualizar las mejores rutas para atender al llamado.

Torre de control: Estos pueden mantener sus operaciones con una prioridad baja y capacidad mínima porque su trabajo empieza cuando las alertas terminan con el análisis de los reportes.

Fase 3: queries y los resultados de análisis de datos.

- Abrir un terminal dentro de la máquina virtual y establecer el directorio de instalación de Hadoop como directorio actual: cd /home/bigdata/hadoop-3.3.6/
- Arrancar HDFS: sbin/start-dfs.sh

```
bigdata@bigdatapc: ~/hadoop-3.3.6
bigdata@bigdatapc: ~/hadoop-3.3.6 128x47
bigdata@bigdatapc:~$  cd /home/bigdata/hadoop-3.3.6/
bigdata@bigdatapc:~/hadoop-3.3.6$ sbin/start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [bigdatapc]
bigdata@bigdatapc:~/hadoop-3.3.6$ S
```

Abrir un terminal nuevo y dirigirse a la carpeta de instalación de Hive: cd /home/bigdata/apache-hive-3.1.3-bin/

- Arrancar el terminal beeline: bin/beeline -u jdbc:hive2://

```
bigdata@bigdatapc:~/hadoop-3.3.6$ cd /home/bigdata/apache-hive-3.1.3-bin/
bigdata@bigdatapc:~/apache-hive-3.1.3-bin$ bin/beeline -u jdbc:hive2://
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/home/bigdata/apache-hive-3.1.3-bin/lib/log4j-slf4j-impl-2.17.1.jar!/org/slf4j/impl/StaticLogg
erBinder.class]
SLF4J: Found binding in [jar:file:/home/bigdata/hadoop-3.3.6/share/hadoop/common/lib/slf4j-reload4j-1.7.36.jar!/org/slf4j/impl/S
taticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Connecting to jdbc:hive2://
Hive Session ID = d4f56d20-1e58-43db-9183-7f08b7b85a22
24/05/29 19:49:03 [main]: WARN session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.ma
nager is set to instance of HiveAuthorizerFactory.
24/05/29 19:49:03 [main]: WARN metastore.ObjectStore: datanucleus.autoStartMechanismMode is set to unsupported value null . Sett
ing it to value: ignored
24/05/29 19:49:04 [main]: WARN util.DriverDataSource: Registered driver with driverClassName=org.apache.derby.jdbc.EmbeddedDrive
r was not found, trying direct instantiation.
24/05/29 19:49:04 [main]: WARN util.DriverDataSource: Registered driver with driverClassName=org.apache.derby.jdbc.EmbeddedDrive
r was not found, trying direct instantiation.
24/05/29 19:49:05 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:05 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:05 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:05 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:05 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:05 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:06 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:06 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:06 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:06 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:06 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
24/05/29 19:49:06 [main]: WARN DataNucleus.MetaData: Metadata has jdbc-type of null yet this is not valid. Ignored
Connected to: Apache Hive (version 3.1.3)
Driver: Hive JDBC (version 3.1.3)
Transaction isolation: TRANSACTION_REPEATABLE_READ
Beeline version 3.1.3 by Apache Hive
0: jdbc:hive2://>
```

- Creacion de tabla. Descripcion de campos:

a) incident_id: Identificador único que permite rastrear cada incidente individualmente.
b) incident_type: Especifica la naturaleza del incidente, lo que ayuda a categorizar y priorizar las respuestas.
c) timestamp: Marca temporal precisa del momento en que ocurrió el incidente, crucial para el análisis temporal y de tendencias.
d) location_x, location_y, location_z: Coordenadas espaciales del incidente, esenciales para la localización y despliegue de equipos de rescate.
e) spaceship_id: Identificador único de la nave espacial involucrada, facilitando el seguimiento y la historia de incidentes por nave.
f) crew_size: Número de personas a bordo, importante para dimensionar la respuesta necesaria.
g) emergency_level: Nivel de gravedad del incidente, que influye en la rapidez y los recursos asignados a la respuesta.

h) response_team_id: Identificador del equipo de respuesta que intervino, permitiendo evaluar su efectividad y tiempos de reacción.
i) response_time_minutes: Tiempo en minutos que tomó la respuesta, un indicador clave de eficiencia.
j) status: Estado actual del incidente, útil para el seguimiento en tiempo real y la gestión de incidentes pendientes.

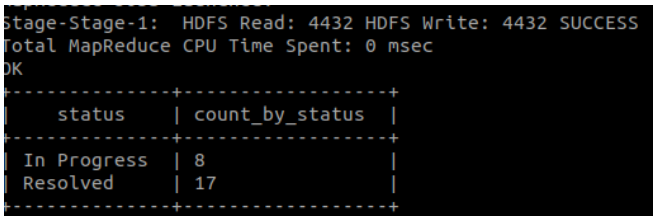
```
0: jdbc:hive2://> CREATE TABLE IF NOT EXISTS incidents (
. . . . . . . . >     incident_id INT,
. . . . . . . . >     incident_type STRING,
. . . . . . . . >     incident_timestamp STRING,
. . . . . . . . >     location_x FLOAT,
. . . . . . . . >     location_y FLOAT,
. . . . . . . . >     location_z FLOAT,
. . . . . . . . >     spaceship_id STRING,
. . . . . . . . >     crew_size INT,
. . . . . . . . >     emergency_level STRING,
. . . . . . . . >     response_team_id STRING,
. . . . . . . . >     response_time_minutes INT,
. . . . . . . . >     status STRING
. . . . . . . . > );
OK
No rows affected (1,361 seconds)
0: jdbc:hive2://>
```

- Insertar datos.

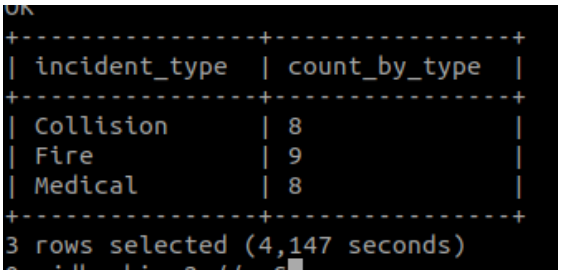
```
0: jdbc:hive2://> INSERT INTO incidents VALUES
. . . . . . . . > (1, 'Fire', '2024-05-01 10:15:00', 123.45, 678.90, 234.56, 'SS-001', 5, 'High', 'RT-01', 15, 'Resolved'),
. . . . . . . . > (2, 'Collision', '2024-05-02 14:20:00', 223.45, 778.90, 334.56, 'SS-002', 8, 'Medium', 'RT-02', 20, 'In Progre
ss'),
. . . . . . . . > (3, 'Medical', '2024-05-03 08:30:00', 323.45, 878.90, 434.56, 'SS-003', 3, 'Low', 'RT-03', 10, 'Resolved'),
. . . . . . . . > (4, 'Fire', '2024-05-04 11:45:00', 423.45, 978.90, 534.56, 'SS-004', 6, 'High', 'RT-04', 25, 'Resolved'),
. . . . . . . . > (5, 'Collision', '2024-05-05 16:10:00', 523.45, 1078.90, 634.56, 'SS-005', 7, 'Medium', 'RT-05', 30, 'In Progr
ess'),
. . . . . . . . > (6, 'Medical', '2024-05-06 09:20:00', 623.45, 1178.90, 734.56, 'SS-006', 4, 'Low', 'RT-06', 15, 'Resolved'),
. . . . . . . . > (7, 'Fire', '2024-05-07 13:50:00', 723.45, 1278.90, 834.56, 'SS-007', 5, 'High', 'RT-07', 20, 'Resolved'),
. . . . . . . . > (8, 'Collision', '2024-05-08 17:25:00', 823.45, 1378.90, 934.56, 'SS-008', 8, 'Medium', 'RT-08', 25, 'In Progr
ess'),
. . . . . . . . > (9, 'Medical', '2024-05-09 07:15:00', 923.45, 1478.90, 1034.56, 'SS-009', 3, 'Low', 'RT-09', 10, 'Resolved'),
. . . . . . . . > (10, 'Fire', '2024-05-10 12:00:00', 1023.45, 1578.90, 1134.56, 'SS-010', 6, 'High', 'RT-10', 15, 'Resolved'),
. . . . . . . . > (11, 'Collision', '2024-05-11 15:40:00', 1123.45, 1678.90, 1234.56, 'SS-011', 7, 'Medium', 'RT-11', 30, 'In Pr
ogress'),
. . . . . . . . > (12, 'Medical', '2024-05-12 09:50:00', 1223.45, 1778.90, 1334.56, 'SS-012', 4, 'Low', 'RT-12', 15, 'Resolved')
,
. . . . . . . . > (13, 'Fire', '2024-05-13 14:35:00', 1323.45, 1878.90, 1434.56, 'SS-013', 5, 'High', 'RT-13', 20, 'Resolved'),
. . . . . . . . > (14, 'Collision', '2024-05-14 18:05:00', 1423.45, 1978.90, 1534.56, 'SS-014', 8, 'Medium', 'RT-14', 25, 'In Pr
ogress'),
. . . . . . . . > (15, 'Medical', '2024-05-15 08:45:00', 1523.45, 2078.90, 1634.56, 'SS-015', 3, 'Low', 'RT-15', 10, 'Resolved')
,
. . . . . . . . > (16, 'Fire', '2024-05-16 13:30:00', 1623.45, 2178.90, 1734.56, 'SS-016', 6, 'High', 'RT-16', 15, 'Resolved'),
. . . . . . . . > (17, 'Collision', '2024-05-17 16:20:00', 1723.45, 2278.90, 1834.56, 'SS-017', 7, 'Medium', 'RT-17', 30, 'In Pr
ogress'),
. . . . . . . . > (18, 'Medical', '2024-05-18 10:10:00', 1823.45, 2378.90, 1934.56, 'SS-018', 4, 'Low', 'RT-18', 15, 'Resolved')
,
. . . . . . . . > (19, 'Fire', '2024-05-19 14:55:00', 1923.45, 2478.90, 2034.56, 'SS-019', 5, 'High', 'RT-19', 20, 'Resolved'),
. . . . . . . . > (20, 'Collision', '2024-05-20 17:35:00', 2023.45, 2578.90, 2134.56, 'SS-020', 8, 'Medium', 'RT-20', 25, 'In Pr
ogress'),
. . . . . . . . > (21, 'Medical', '2024-05-21 07:25:00', 2123.45, 2678.90, 2234.56, 'SS-021', 3, 'Low', 'RT-21', 10, 'Resolved')
,
. . . . . . . . > (22, 'Fire', '2024-05-22 12:40:00', 2223.45, 2778.90, 2334.56, 'SS-022', 6, 'High', 'RT-22', 15, 'Resolved'),
. . . . . . . . > (23, 'Collision', '2024-05-23 15:55:00', 2323.45, 2878.90, 2434.56, 'SS-023', 7, 'Medium', 'RT-23', 30, 'In Pr
ogress'),
. . . . . . . . > (24, 'Medical', '2024-05-24 09:35:00', 2423.45, 2978.90, 2534.56, 'SS-024', 4, 'Low', 'RT-24', 15, 'Resolved')
,
. . . . . . . . > (25, 'Fire', '2024-05-25 13:45:00', 2523.45, 3078.90, 2634.56, 'SS-025', 5, 'High', 'RT-25', 20, 'Resolved');
24/05/29 20:01:26 [HiveServer2-Background-Pool: Thread-59]: WARN ql.Driver: Hive-on-MR is deprecated in Hive 2 and may not be av
ailable in the future versions. Consider using a different execution engine (i.e. spark, tez) or using Hive 1.X releases.
```

- Consultas

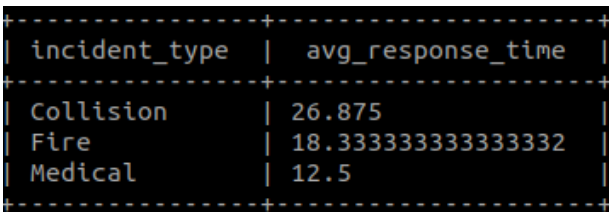
Contar el número de incidentes por estado. SELECT status, COUNT(*) AS count_by_status FROM incidents GROUP BY status;



Contar el número de incidentes por tipo. SELECT incident_type, COUNT(*) AS count_by_type FROM incidents GROUP BY incident_type;



Obtener el promedio de tiempo de respuesta de incidentes por tipo. SELECT response_team_id, AVG(response_time_minutes) AS avg_response_time_by_team FROM incidents GROUP BY response_team_id;



Contar el número de incidentes por equipo de respuesta. SELECT response_team_id, COUNT(*) AS count_by_team FROM incidents GROUP BY response_team_id;

response_team_id	count_by_team
RT-01	1
RT-02	1
RT-03	1
RT-04	1
RT-05	1
RT-06	1
RT-07	1
RT-08	1
RT-09	1
RT-10	1
RT-11	1
RT-12	1
RT-13	1
RT-14	1
RT-15	1
RT-16	1
RT-17	1
RT-18	1
RT-19	1
RT-20	1
RT-21	1
RT-22	1
RT-23	1
RT-24	1
RT-25	1

Obtener el promedio de tiempo de respuesta de incidentes por equipo de respuesta. SELECT response_team_id, AVG(response_time_minutes) AS avg_response_time_by_team FROM incidents GROUP BY response_team_id;

response_team_id	avg_response_time_by_team
RT-01	15.0
RT-02	20.0
RT-03	10.0
RT-04	25.0
RT-05	30.0
RT-06	15.0
RT-07	20.0
RT-08	25.0
RT-09	10.0
RT-10	15.0
RT-11	30.0
RT-12	15.0
RT-13	20.0
RT-14	25.0
RT-15	10.0
RT-16	15.0
RT-17	30.0
RT-18	15.0
RT-19	20.0
RT-20	25.0
RT-21	10.0
RT-22	15.0
RT-23	30.0
RT-24	15.0
RT-25	20.0