

```
36 'dragover',
37 'drop'
38 ];
39 events.forEach(e => {
40   fileDropZone.addEventListener(e, (ev) => {
41     ev.preventDefault();
42     if (ev.type === 'dragenter') {
43       fileDropZone.classList.add('solid-border');
44     }
45     if (ev.type === 'dragleave') {
46       fileDropZone.classList.remove('solid-border');
47     }
48     if(ev.type === 'drop') {
49       fileDropZone.classList.remove('solid-border');
50       handleFiles(ev.dataTransfer.files)
51       .map(values => values.map(tag => {
52         document.querySelector(`[class="${tag}"]`)
53         .setAttribute('class', 'border rounded ingreyed')
54         .appendChild(tag)
55       })
56     )
57   }
58 })
59 
```

webMethods Clustering Guide - 10.11 (API + EAI(IS))

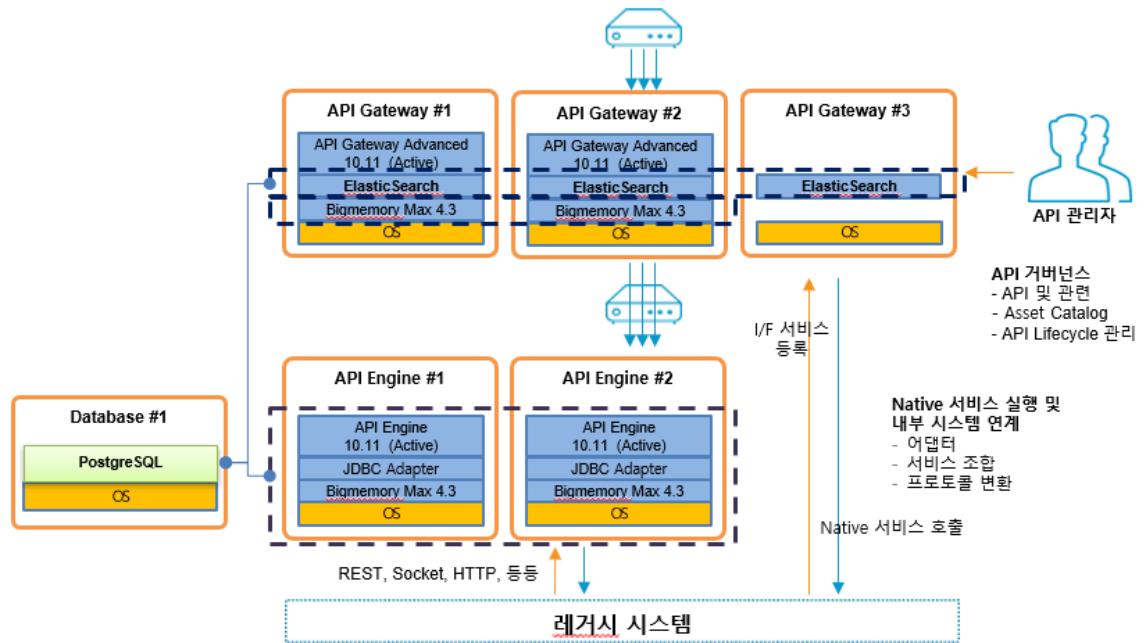
• References

- <https://rastalion.me/elasticsearch-엘라스틱-서치-7-설치-및-cluster-구성하기/>
(ElasticSearch 클러스터링 관련)
- <https://itstarter.tistory.com/779> (Quorum 클러스터 & 스플릿 브레인 현상)
- <https://www.nextree.co.kr/p3151/amp/> (Terracotta Server Array - 분산 캐시 클러스터링)
- https://documentation.softwareag.com/webmethods/api_gateway/yai10-11/10-11_API_Gateway_webhelp/index.html#page/api-gateway-integrated-webhelp%2Fco-cluster_deployment.html%23 (SoftwareAG 공식문서)

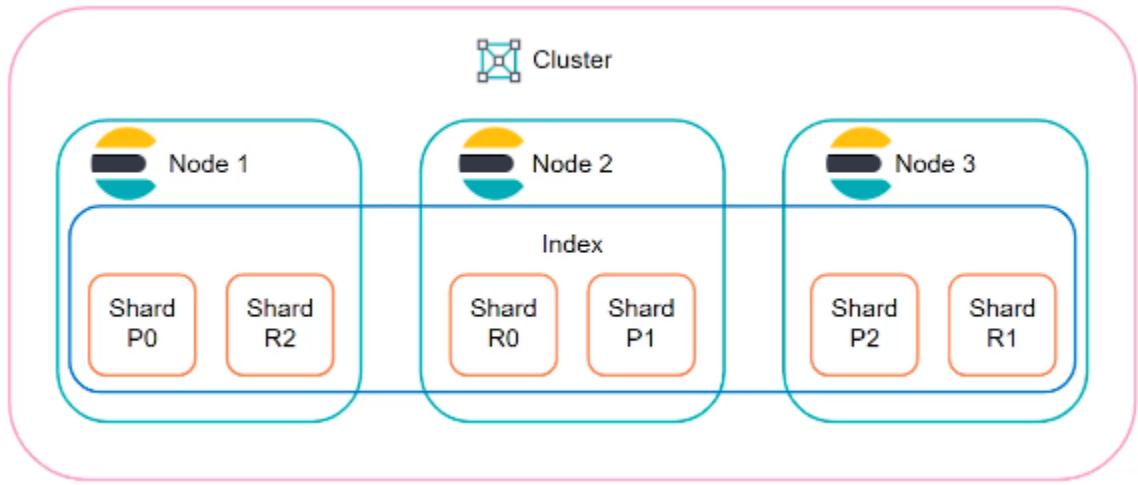
• 클러스터링이란

- 따로따로 작동하는 여러 컴퓨터를 논리적으로 결합하여 전체를 한 대의 컴퓨터처럼 이용할 수 있게 하는 시스템 구축 기술
- 가용성과 성능을 높이며, 보통 webMethods 에서는 고가용성(HA) 을 위한 클러스터 구성을 의미

- SoftwareAG API Management 아키텍처 예시

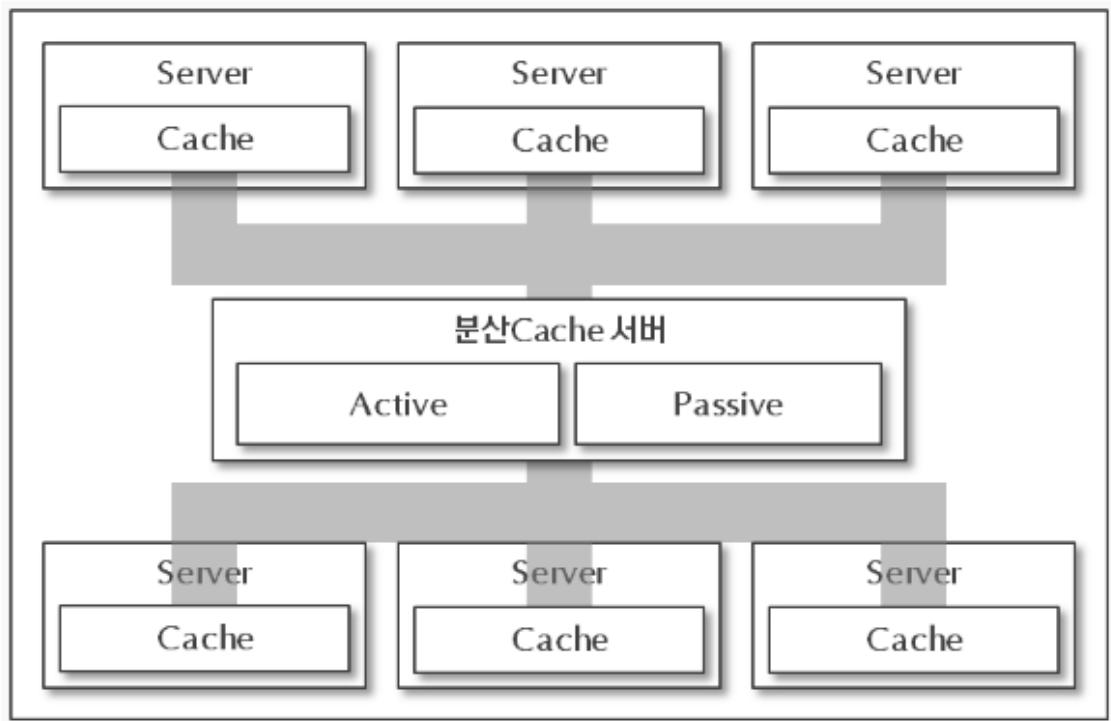


- 제품 구성
 - API Gateway : API Gateway + Integration Server (API 용)
 - API Engine : Integration Server (EAI 용)
- Elasticsearch는 스플릿 브레인 현상을 피하기 위해 Quorum 클러스터 구성, 3대 이상의 허수로 구성 (N중화, Master - Slave 구조)
- Integration Server는 Terracotta 서버 (Bigmemory Max)를 사용한 분산 캐시 클러스터링 구성 (2중화, Active - Passive 구조)
- Elasticsearch Clustering (N중화)



- API Gateway 내부 저장소 클러스터링 구성
- 훌수로 구성하는것을 원칙으로 하며 보통 3중화 구성
- **Terracotta Server Array (TSA) (BigMemory Max)**

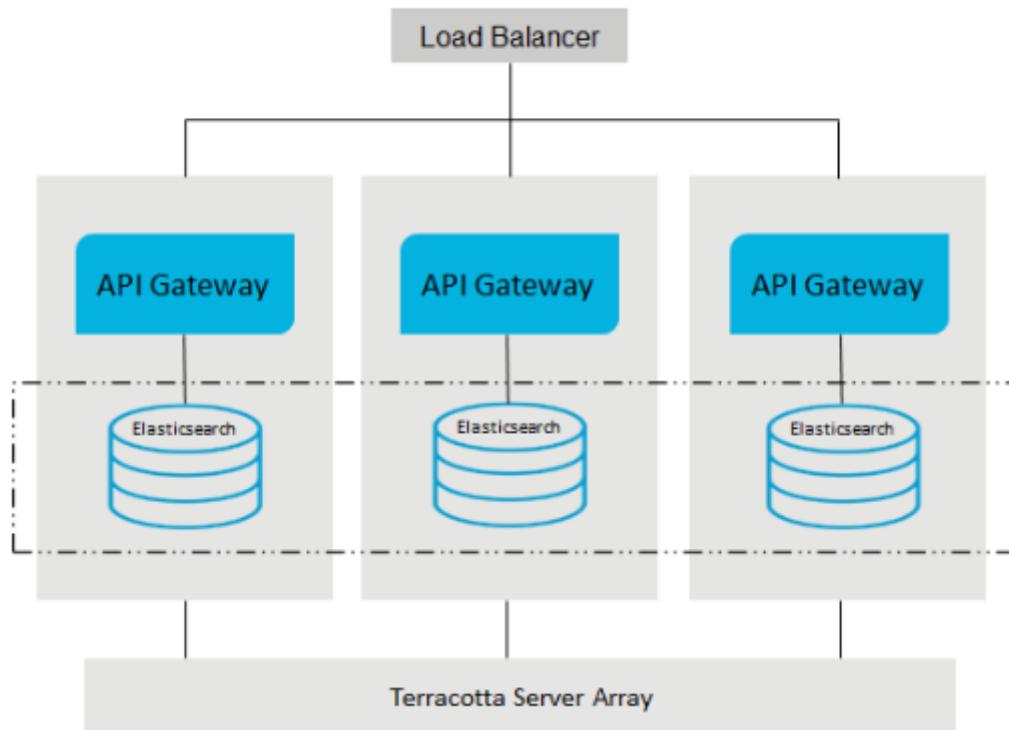
Hub and Spoke 분산 캐시 아키텍처



- Terracotta는 각 캐시 노드들의 Hub 역할을 하는 분산 캐시 서버
- Ehcache+Terracotta의 조합으로 여러 캐시 노드의 동기화를 지원
- Hub-Spoke 구조이기 때문에 캐시의 동기화를 위한 부하는 Terracotta 분산캐시 서버에 위임
- Terracotta Server는 여러대로 이중화 구성이 가능
- Active-Passive 모드로 서비스를 하고, 장애발생 시 자동으로 Fault Tolerance 기능을 제공
- webMethods 에서는 패키지 관련, 설정 관련 데이터 동기화
- GUI 환경 Clustering On/Off 기능 제공

- **TSA 를 사용하는 Clustering**

TSA를 사용하는 클러스터 환경



3 Node 설치 시

- Terracotta 이중화 → Elasticsearch 3중화 순서로 설정하는게 편리
- **Terracotta Server Array 구성 Guide (이중화) - Integration Server 이중화**
 - BigMemory Max 4.3 설치 필요

- **STEP 1.** 각 Node 의 Host명 설정
 - /etc/hosts 파일 수정 예시

```

127.0.0.1    localhost localhost.localdomain localhost4 localhost4.localdomain4
::1          localhost localhost.localdomain localhost6 localhost6.localdomain6

## API Gateway Nodes
192.168.1.x   node1
192.168.1.x   node2
192.168.1.x   node3
~
```

각 노드 호스트명을 node1, node2, node3 같은 호스트명으로 구성 (각 노드에 동일하게)

- **STEP 2. API Gateway 화면에서 Clustering Enabled & Terracotta License 설정 확인**
 - Administration > Clustering
 - Cluster name 은 모든 노드 동일하게 설정해야 함

Clustering configuration
Configure API Gateway to work in cluster mode. [?](#)

Enabled

Select cluster type
 [Terracotta](#)

Cluster name*	Session timeout (in minutes)*	Action on clustering failure
Clustering_TEST	60	Start as stand alone API Gateway

Terracotta server array urls*

	+ Add
Terracotta server array urls	Action
192.168.1.54:9510	
192.168.1.164:9510	
192.168.1.227:9510	

[Cancel](#) [Save](#)

■ Terracotta License 설정 확인

- Terracotta License 는 설치 시 또는 해당 설정 디렉토리 (`{SAG_HOME}/common/conf/`)에 업로드 후 SAVE해야 함

Application logs

▼ License

Configuration

Details

Clustering

Callback processor

Configure license
Configure the API Gateway and Terracotta license file path here. [?](#)

API Gateway license file

Terracotta license file

[Cancel](#) [Save](#)

- **STEP 3.** Terracotta 설정 파일 수정 (tc-config.xml) & Terracotta Server Start
 - tc-config.xml 파일 수정 후 `{SAG_HOME}/Terracotta/server/bin` 에 해당 파일 업로드
 - https://documentation.softwareag.com/webmethods/api_gateway/yai10-11/10-11_API_Gateway_webhelp/index.html#page/api-gateway-integrated-webhelp%2Fco-terracotta_server_array_configuration.html%23 (tc-config 파일 예시)

```

<?xml version="1.0" encoding="UTF-8" ?>
<tc:tc-config xmlns:tc="http://www.terracotta.org/config" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tc-properties>
    <property name="11.cachemanager.enabled" value="true"/>
    <property name="12.nha.dirtydb.autoDelete" value="true"/>
    <property name="12.nha.dirtydb.rolling" value="2"/>
    <property name="Logging.maxLogFileSize" value="1024"/>
    <property name="12.llreconnect.enabled" value="true"/>
    <property name="12.llreconnect.timeout.millis" value="2000"/>
  </tc-properties>
  <servers>
    <server host="192.168.1.160" name="datacenter01" bind="0.0.0.0">
      <data>/webM/APIGW/Terracotta/server/datas/server-datas</data>
      <logs>/webM/APIGW/Terracotta/server/datas/server-logs</logs>
      <index>/webM/APIGW/Terracotta/server/datas/server-index</index>
      <tsa-port bind="192.168.1.160">9510</tsa-port>
      <jmx-port bind="192.168.1.160">9520</jmx-port>
      <tsa-group-port bind="192.168.1.160">9530</tsa-group-port>
      <management-port bind="192.168.1.160">9540</management-port>
    </server>
    <server host="192.168.1.218" name="datacenter02" bind="0.0.0.0">
      <data>/webM/APIGW/Terracotta/server/datas/server-datas</data>
      <logs>/webM/APIGW/Terracotta/server/datas/server-logs</logs>
      <index>/webM/APIGW/Terracotta/server/datas/server-index</index>
      <tsa-port bind="192.168.1.218">9510</tsa-port>
      <jmx-port bind="192.168.1.218">9520</jmx-port>
      <tsa-group-port bind="192.168.1.218">9530</tsa-group-port>
    </server>
  </servers>
  <offheap>
    <enabled>false</enabled>
    <maxDataSize>2048m</maxDataSize>
  </offheap>
</tc-config>

```

<server name =”서버명“>

- 테라코타 서버 시작 : nohup ./start-tc-server.sh -f
`/{SAG_HOME}/Terracotta/server/bin/tc-config.xml -n {server_name} &` 로 실행
 - server_name은 tc-config 파일 내 server name
 - nohup + & 명령은 세션이 끊겨도 백그라운드실행을 위한 커맨드, nohup 과 & 를 빼고 테스트 하며 에러로그를 확인 후 정상적으로 Clustering 되었을 때 실행하는 편이 좋음
- 테라코타 서버 중지 : ./stop-tc-server.sh
- 테라코타 서버 상태 확인 : ./server-stats -f tc-config.xml
- Elasticsearch Clustering Guide (3중화) - API Gateway 3중화
 - STEP 1. 각 노드의 `/{SAG_HOME}/InternalDataStore/data` 폴더 삭제
 - data 폴더 안에는 설치 & 기동 시 자동으로 생성 된 index 들과 데이터들이 존재
 - 각 노드의 데이터가 다르면 동기화가 안되므로, 동기화 시키려면 같은 Index Id 값이 필요하기에 삭제

```

drwxrwxr-x. 2 sol sol 6 Jun 16 21:44 archives
drwxrwxr-x. 2 sol sol 4096 Jul 29 02:36 bin
drwxrwxr-x. 2 sol sol 168 Jun 16 21:44 config
drwxrwxr-x. 3 sol sol 19 Jun 16 21:44 data
drwxrwxr-x. 3 sol sol 4096 Jun 16 21:37 lib
drwxrwxr-x. 2 sol sol 12288 Sep 4 21:30 logs
drwxrwxr-x. 59 sol sol 4096 Jun 16 21:37 modules
drwxrwxr-x. 3 sol sol 25 Jun 16 21:37 plugins
drwxrwxr-x. 3 sol sol 29 Jul 29 02:36 temp
[sol@localhost InternalDataStore]$ pwd
/webM/InternalDataStore
[sol@localhost InternalDataStore]$ rm -rf /data

```

- **STEP 2. elasticsearch.yml 파일 수정**
(<SAG_HOME>InternalDataStore/config)
 - Elasticsearch Config 파일 수정

host_name:port_name

If there are three nodes in the cluster, the value in the **discovery.seed_hosts** property will be like the example given here:

discovery.seed_hosts: ["node1:9340", "node2:9340", "node3": "9340"]

The names of all nodes appear in the **cluster.initial_master_nodes** property. The node name displayed in this property is same as seen in the **node.name** property.

Sample configuration of a node is as follows:

```

cluster.name:"SAG_EventDataStore"
node.name: node1
path.logs: SAG_root\InternalDataStore/logs
network.host:0.0.0.0
http.port:9240
discovery.seed_hosts: ["hostname1:9340", "hostname2:9340", "hostname3:9340"]
transport.tcp.port:9340
path.repo:['SAG_root\InternalDataStore/archives']
cluster.initial_master_nodes:["node1", "node2", "node3"]

```

The specified nodes are clustered

- 수정 예시 (227,164,53 서버의 클러스터링) - 227 서버의 Config 파일

```

1 cluster.name: SAG_EventDataStore
2 node.name: 192.168.1.227
3 path.logs: /webM2/internal/DataStore/logs
4 network.host: 0.0.0.0
5
6 http.port: 9240
7
8 discovery.seed_hosts: ["192.168.1.227:9340","192.168.1.164:9340","192.168.1.54:9340"]
9 transport.tcp.port: 9340
10 path.repo: [/webM2/internalDataStore/archives]
11
12 cluster.initial_master_nodes: ["192.168.1.227","192.168.1.164","192.168.1.54"]
13

```

해당 테스트에서는 편의상 Host명, Node명을 IP명과 동일하게 설정했습니다. (hostnamectl set-hostname 192.168.1.227 & etc/hosts 파일에서 node1 = 192.168.1.227로 구성)

■ 설정 상세

- **cluster.name:** 엘라스틱 서치 클러스터의 이름 = SAG_EventDataStore로 각 노드 전부 동일하게 구성
- **node.name:** 엘라스틱 서치의 노드 이름, \${HOSTNAME} 으로 서버의 호스트네임을 가져다 쓸 수 있으며, Custom Node 명으로 설정 가능
- **path.logs:** 엘라스틱 서치의 로그가 기록되는 경로
- **network.host:** 특정 IP만 접근 허용하기 위한 설정 (0.0.0.0 = 모든 IP 허용), bind_host와 public_host를 동시에 설정에 적용 가능
- **http.port:** Elasticsearch HTTP 호출(API 호출) 을 위한 포트번호
- **discovery.seed_hosts:** 엘라스틱 서치 클러스터에 포함 되는 노드들의 배열입니다. ["192.168.0.76", "192.168.0.90", "192.168.0.98", "192.168.0.101"] 이런식으로 기입
- **transport.tcp.port:** Elasticsearch TCP 호출을 위한 포트 번호 설정
- **path.repo:** 인덱스를 백업하기 위한 스냅샷 경로
- **cluster.initial_master_nodes:** 엘라스틱 서치의 클러스터의 마스터 노드 그룹을 지정하는 배열을 삽입, Master 노드와 후보 노드들을 기입

- **STEP 3. Server Restart & Elastic Search Health 체크**
 - 각 Gateway 의 API가 동기화 되어있는지 확인

The image displays three vertically stacked screenshots of the 'Manage APIs' interface from the webMethods API Gateway. Each screenshot shows a table with columns for 'Name', 'Description', and 'Version'. A single row is selected, showing 'ES_Test' as the name, 'REST' as the type, and '1.0' as the version. The first two screenshots also show a 'Type' filter dropdown with 'REST' selected. The third screenshot shows the 'Manage APIs' header and a note 'Create and manage your APIs.' Below the table, there is a text input field containing the Korean character '예시' (Example).

Add filter	Name	Description	Version
Type <input checked="" type="checkbox"/> REST	REST ES_Test		1.0

Version: 10.5.0.9.362

Add filter	Name	Description	Version
Type <input checked="" type="checkbox"/> REST <input type="checkbox"/> SOAP	REST ES_Test		1.0

Version: 10.5.0.0.579

Add filter	Name	Description	Version
Type <input type="checkbox"/> REST	REST ES_Test		1.0

Manage APIs
Create and manage your APIs. ?

예시

- Elasticsearch 상태 확인 (API) : [GET]
http://node_ip:9240/_cluster/health 호출 (POSTMAN)

http://192.168.1.227:9240/_cluster/health

GET http://192.168.1.227:9240/_cluster/health

Params Authorization Headers (6) Body Pre-request

Query Params

KEY
Key

Body Cookies Headers (3) Test Results

Pretty Raw Preview Visualize JSON

```
1 [ {  
2   "cluster_name": "SAG_EventDataStore",  
3   "status": "green",  
4   "timed_out": false,  
5   "number_of_nodes": 3,  
6   "number_of_data_nodes": 3,  
7   "active_primary_shards": 104,  
8   "active_shards": 210,  
9   "relocating_shards": 0,  
10  "initializing_shards": 0,  
11  "unassigned_shards": 0,  
12  "delayed_unassigned_shards": 0,  
13  "number_of_pending_tasks": 0,  
14  "number_of_in_flight_fetch": 0,  
15  "task_max_waiting_in_queue_millis": 0,  
16  "active_shards_percent_as_number": 100.0  
} ]
```