**Elasticsearch ingestion pipeline**

1. Create a source index for a dataset from which we have to pass the data. In the following figure we have pink blocks depicts the data index. In our case it will be ***ip2loc*** index.

PUT /ip2loc

{

"mappings": {

"properties": {

"ipranges": {

"type": "ip\_range"

},

"isocode": {

"type": "keyword"

},

"countryname": {

"type": "keyword"

},

"province": {

"type": "keyword"

},

"city": {

"type": "keyword"

},

"latitude": {

"type": "float"

},

"longitude": {

"type": "float"

},

"isp": {

"type": "keyword"

}

}

}

}

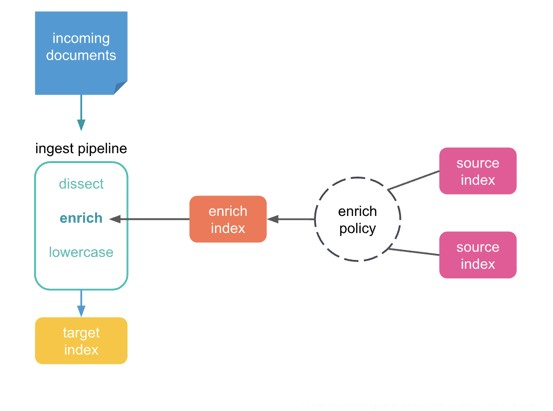


Fig: Data enrichment in Elasticsearch with the ingest pipeline

1. In the next step we have to insert the data using elasticsearch bulk api or any programing language api (python, java, c# etc.) In my case I inserted data using bulk api of elasticsearch in devtool of kibana as well as using python.

POST ip2loc/\_bulk

{"index":{"\_id":0}}

{"isocode":"-","countryname":"-","province":"-","city":"-","latitude":0.0,"longitude":0.0,"ipranges":"0.0.0.0/8", "isp": "Broadcast RFC1700"}

{"index":{"\_id":1}}

{"isocode":"US","countryname":"United States of America","province":"California","city":"Los Angeles","latitude":34.052859,"longitude":-118.24357,"ipranges":"1.0.0.0/24", "isp": "APNIC and CloudFlare DNS Resolver Project"}

{"index":{"\_id":2}}

{"isocode":"CN","countryname":"China","province":"Fujian","city":"Fuzhou","latitude":26.06139,"longitude":119.30611,"ipranges":"1.0.1.0/24","isp": "ChinaNet Fujian Province Network"}

{"index":{"\_id":3}}

{"isocode":"CN","countryname":"China","province":"Fujian","city":"Fuzhou","latitude":26.06139,"longitude":119.30611,"ipranges":"1.0.2.0/23","isp": "Fortinet Japan"}

{"index":{"\_id":4}}

{"isocode":"AU","countryname":"Australia","province":"Tasmania","city":"Glebe","latitude": -42.87471, "longitude": 147.32813,"ipranges":"1.0.4.0/24", "isp": "GTelecom Pty Ltd"}

Or using python script which is ***BulkInsertElastic.py. attached here in this directory***

1. After data insertion in source index, we need to create enrich policy. In this policy we have to mention that which is the source index, match field and enrich fields.

PUT \_enrich/policy/ip2loc\_policy

{

"range": {

"indices": "ip2loc",

"match\_field": "ipranges",

"enrich\_fields": ["countryname", "province"]

}

}

We can check all policies using

GET /\_enrich/policy

And delete specific policy using

DELETE /\_enrich/policy/ip2loc\_policy

1. In the fourth step we have to execute enrich policy.

POST /\_enrich/policy/ip2loc\_policy/\_execute?wait\_for\_completion=false

In this step we have to wait for enrichment index for populating. You can check it using

GET /\_tasks/pO7K7r3cR02tUMWLKnubcA:1227945

GET \_tasks?actions=policy\*

GET .enrich-\*

1. Once we have all enrichment index created and data is inside this enrichment index. We are good to go with creation of ingestion pipeline.

PUT /\_ingest/pipeline/ip2\_loc\_lookups

{

"description": "\_description",

"processors": [

{

"enrich": {

"policy\_name": "ip2loc\_policy",

"field": "src\_ip",

"target\_field": "geo",

"max\_matches": "1"

}

}

]

}

In our case, ip2\_loc\_lookups is the ingestion pipeline. We are checking the src\_ip field in the incoming document and appending geo field with countryname and provice as enrichment field in the target index.

1. Finally we can check our ingestion pipeline using simulate or default pipeline for our target index. Something like this:

POST /\_ingest/pipeline/ip2\_loc\_lookups/\_simulate

{

"docs": [

{

"\_source": {

"src\_ip": "0.0.0.1"

}

},

{

"\_source": {

"src\_ip": "2.2.2.8"

}

}

]

}

OR

PUT sampleIndex

{

"template": {

"settings": {

"index.default\_pipeline": " ip2\_loc\_lookups",

}

}

}