Extending Elasticsearch

Writing plugins, fast and easy!

Alexander Reelsen
@spinscale
alex@elastic.co

elastic

Elasticsearch in 10 seconds

- Search Engine (FTS, Analytics, Geo), real-time
- Distributed, scalable, highly available, resilient
- Interface: HTTP & JSON
- Centrepiece of the Elastic Stack (Kibana, Logstash, Beats, APM, ML, App/Site/Enterprise Search)
- Uneducated guess: Tens of thousands of clusters worldwide, hundreds of thousands of instances

Agenda

- Overview
- Building
- Testing
- Releasing
- Extension points

Overview

Requirements, gotchas and some details

elastic

Overview

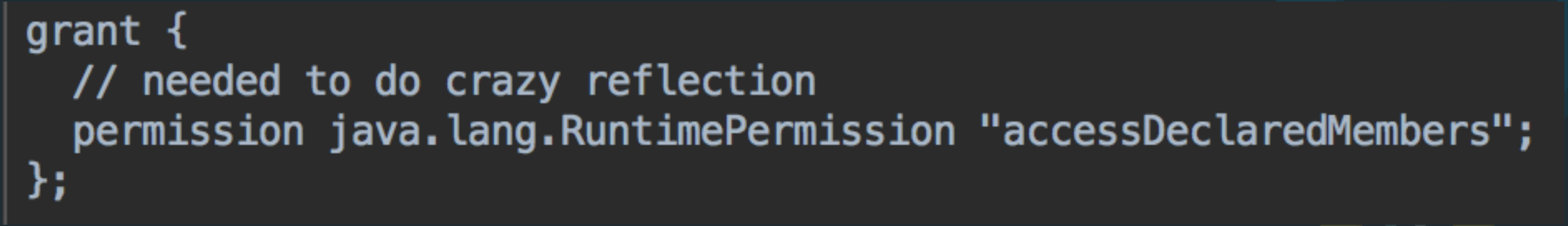
- Plugins are zip files
- own jars/dependencies
- loaded with its own classloader
- own security permissions

Security Manager? Permissions? But why?!

- Sandbox your java application
- Prevent certain calls by your application
- Policy file grants permissions
  - `FilePermission` (read, write)
  - `SocketPermission` (connect, listen, accept)
  - `URLPermission`, `PropertyPermission`, ...

Security Manager? Permissions? But why?!

- Each plugin needs to be packaged with a policy that grants certain operations



```
grant {
  // needed to do crazy reflection
  permission java.lang.RuntimePermission "accessDeclaredMembers";
};
```

Building

```
gradle clean assemble
```

elastic

Overview

- A plugin needs to be released for an Elasticsearch patch release
- Use the gradle plugin! Batteries included, lots of them
- Integration tests, license checks, checkstyle, notice files

Testing

```
gradle clean check
```

Overview

- Unit tests: `ESTestCase`
- Integration tests: YAML
  - Download ES, install plugin, start ES, run tests against HTTP

ESTestCase

- Randomized testing
- Logger
- Thread leak detector
- Deprecation warnings detection
- Test method names have to begin with `void test...()`
- `assertBusy/awaitBusy`

Extension points

```
class MyPlugin extends Plugin
```

elastic

Overview

- `ActionPlugin`: Implement own actions, REST endpoints
- `AnalysisPlugin`: Add custom parts to analysis chain
- `ClusterPlugin`: Implement own shard allocation logic
- `DiscoveryPlugin`: Add (often API based) host discovery
- `IngestPlugin`: Custom ingest processors
- `MapperPlugin`: Custom mapping data types
- `NetworkPlugin`: Network implementations (i.e. netty replacement)
- `RepositoryPlugin`: Snapshot/restore repository implementations
- `ScriptPlugin`: Script Engines, Script Contexts
- `SearchPlugin`: Queries, Highlighter, Suggester, Aggregations, Rescoring, Search Extensions
- `ReloadablePlugin`: Ensure a plugin can reload its state

Overview

- main class: **`org.elasticsearch.plugins.Plugin`**
- settings registration & filter
- custom cluster state metadata
- cluster state listener
- index modules (search listener, index event listener)

Plugin Loading

```
[INFO ][o.e.p.PluginsService] [node] loaded module [lang-mustache]
```

elastic

Overview

- `PluginService`
- check for modules & plugins directory
- check jar hell
- check version
- create classloader
- reload lucene SPI
- load Plugin class
- instantiate Plugin class

Show me the code!

Ingest lang-detect processor

```
PUT _ingest/pipeline/langdetect-pipeline
{
 "processors": [
     {
       "langdetect" : {
         "field" : "my_field",
         "target_field" : "language"
       }
     }
   ]
}
```

Ingest lang-detect processor

```
PUT /my-index/my-type/1?pipeline=langdetect-pipeline
{
  "my_field" : "This is hopefully an english text, that will be
detected."
}

GET /my-index/my-type/1
{
  "my_field" : "This is hopefully an english text, that will be
detected.",
  "language": "en"
}
```

Custom query parser

- Ecommerce use-case
- clean data
- non expert searches

Custom query parser

nike running hoodie xl

Custom query parser

nike running hoodie xl

Custom query parser

$\underbrace{\text{nike}}_{\texttt{brand}}\ \text{running hoodie}\ \underbrace{\text{xl}}_{\texttt{size}}$

Custom query parser

```
PUT my_products
{
  "settings": {
    "index.queryparser.values" : {
      "brands" : [ "nike", "adidas", "puma", "salomon" ],
      "size" : [ "m", "xl", "l", "s", "xs", "xxs", "xxl" ]
    }
  }
}
```

Custom query parser

```
GET my_products/_search
{
  "query": {
    "custom" : {
      "title": "nike running hoodie XL"
    }
  }
}
```

elastic

DEMO

Summary

When is your plugin ready again?

elastic

Summary

- Writing own plugins is not that hard
- Testing plugins is incredibly easy with `gradle`
- Release via github or sonatype

Thanks for listening!

Questions?

Alexander Reelsen
@spinscale
alex@elastic.co

elastic

Resources

- https://github.com/elastic/elasticsearch/
- https://github.com/elastic/elasticsearch/tree/7.1/plugins/examples
- https://github.com/spinscale/elasticsearch-ingest-opennlp
- https://github.com/spinscale/elasticsearch-ingest-langdetect
- https://github.com/spinscale/cookiecutter-elasticsearch-ingest-processor

Thanks for listening!

Questions?

Alexander Reelsen
@spinscale
alex@elastic.co

elastic