Indexing

Kube-burner can index the collected metrics metrics into a given indexer.

Indexers

Configured in the `indexerConfig` object, they can be tweaked by the following parameters:

Option	Description	Type	Default
`enabled`	Enable indexing	Boolean	false
`type`	Type of indexer	String	“”

!!! Note At the moment `elastic` and `local` are the only supported indexers

Elastic

This indexer send collected documents to Elasticsearch 7 instances.

The `elastic` indexer can be configured by the parameters below:

Option	Description	Type	Default
`esServers`	List of ES instances	List	“”
`defaultIndex`	Default index to send the prometheus metrics into	String	“”
`insecureSkipVerify`	TLS certificate verification	Boolean	false

!!!Info It’s possible to index documents in an authenticated ES instance using the notation `http(s)://[username]:[password]@[address]:[port]` in the `esServers` parameter.

Local

This indexer writes collected metrics to local files.

The `local` indexer can be configured by the parameters below:

Option	Description	Type	Default
`metricsDirectory`	Collected metric will be dumped here.	String	collected-metrics
`createTarball`	Create metrics tarball	Boolean	false

Option	Description	Type	Default
`tarballName`	Name of the metrics tarball	String	kube-burner-metrics.tgz

Job Summary

When an indexer is configued, at the end of each job, a document holding the job summary is indexed. This is useful to identify the parameters the job was executed with:

This document looks like:

```json
{
  "timestamp": "2020-11-13T13:55:31.654185032+01:00",
  "uuid": "bdb7584a-d2cd-4185-8bfa-1387cc31f99e",
  "metricName": "jobSummary",
  "elapsedTime": 8.768932955,
  "jobConfig": {
    "jobIterations": 10,
    "jobIterationDelay": 0,
    "jobPause": 0,
    "name": "kubelet-density",
    "objects": [
      {
        "objectTemplate": "templates/pod.yml",
        "replicas": 1,
        "inputVars": {
          "containerImage": "gcr.io/google_containers/pause-amd64:3.0"
        }
      }
    ],
    "jobType": "create",
    "qps": 5,
    "burst": 5,
    "namespace": "kubelet-density",
    "waitFor": null,
    "maxWaitTimeout": 43200000000000,
    "waitForDeletion": true,
    "podWait": false,
    "waitWhenFinished": true,
    "cleanup": true,
    "namespacedIterations": false,
    "verifyObjects": true,
    "errorOnVerify": false
  }
}
```

Metric exporting & importing

When using the `local` indexer, it's possible to dump all the collected metrics to a tarball, so we import them later, this is useful in disconnected environments, where kube-burner doesn't have direct access to a ElasticSearch instance. Metrics exporting can be configured by `createTarball` field of the indexer config as noted in the local indexer.

The metric exporting feature is available in by using the `init` and `index` subcommands.

Once we've enabled it, a tarball (`kube-burner-metrics-<timestamp>.tgz`) containing all metrics will be generated in the current working directory. This tarball can be imported and indexed by kube-burner with the `import` subcommand. For example:

```
$ kube-burner/bin/kube-burner import --config kubelet-config.yml --tarball kube-burner-metri
INFO[2021-06-23 11:39:40]  Creating indexer: elastic
INFO[2021-06-23 11:39:42] Importing tarball kube-burner-metrics-1624441857.tgz
INFO[2021-06-23 11:39:42] Importing metrics from doc.json
INFO[2021-06-23 11:39:43] Indexing [1] documents in kube-burner
INFO[2021-06-23 11:39:43] Successfully indexed [1] documents in 208ms in kube-burner
```

Scraping from multiple endpoints

It's possible to scrape from multiple prometheus endpoints and send the results to the target indexer with the `init` and `index` subcommands. This feature is configured by the flag `--metrics-endpoint`, which points to a YAML file with the required configuration.

A valid file provided to the `--metrics-endpoint` would look like this.

```
- endpoint: http://localhost:9090  # This is one of the Prometheus endpoints
  token: <token>                   # Authentication token
  profile: metrics.yaml            # Metrics profile to use in this target
  alertProfile: alerts.yaml        # Alert profile, optional
- endpoint: http://remotehost:9090 # Another Prometheus endpoint
  token: <token>
  profile: metrics.yaml
```

!!!note The configuration provided by the `--metrics-endpoint` flag has precedence over the parameters specified in the config file. The `profile` and `alertProfile` parameters are optional. If not provided they will be taken from the CLI flags.