Purpose

- `import-boss` enforces import restrictions against all pull requests submitted to the k/k repository. There are a number of `.import-restrictions` files that in the k/k repository, all of which are defined in `YAML` (or `JSON`) format.

How does it work?

- When a directory is verified, `import-boss` looks for a file called `.import-restrictions`. If this file is not found, `import-boss` will go up to the parent directory until it finds this `.import-restrictions` file.
- Adding `.import-restrictions` files does not add them to CI runs. They need to be explicitly added to `hack/verify-import-boss.sh`. Once an `.import-restrictions` file is added, all of the sub-packages of this file's directory are added as well.

What are Rules?

- If an `.import-restrictions` file is found, then all imports of the package are checked against each `rule` in the file. A `rule` consists of three parts:
  - A `SelectorRegexp`, to select the import paths that the rule applies to.
  - A list of `AllowedPrefixes`
  - A list of `ForbiddenPrefixes`
- An import is allowed if it matches at least one allowed prefix and does not match any forbidden prefixes. An example `.import-restrictions` file looks like this:

```
{
  "Rules": [
    {
      "SelectorRegexp": "k8s[.]io",
      "AllowedPrefixes": [
        "k8s.io/gengo/examples",
        "k8s.io/kubernetes/third_party"
      ],
      "ForbiddenPrefixes": [
        "k8s.io/kubernetes/pkg/third_party/deprecated"
      ]
    },
    {
      "SelectorRegexp": "^unsafe$",
      "AllowedPrefixes": [
      ],
      "ForbiddenPrefixes": [
        ""
      ]
    }
  ]
}
```

- Take note of `"SelectorRegexp": "k8s[.]io"` in the first block. This specifies that we are applying these rules to the `"k8s.io"` import path.
- The second block explicitly matches the "unsafe" package, and forbids it ("" is a prefix of everything).

What are Inverse Rules?

- In contrast to non-inverse rules, which are defined in importing packages, inverse rules are defined in imported packages.
- Inverse rules allow for fine-grained import restrictions for "private packages" where we don't want to spread use inside of kubernetes/kubernetes.
- If an `.import-restrictions` file is found, then all imports of the package are checked against each `inverse rule` in the file. This check will continue, climbing up the directory tree, until a match is found and accepted.
- Inverse rules also have a boolean `transitive` option. When this option is true, the import rule is also applied to `transitive` imports.
    - `transitive` imports are dependencies not directly depended on by the code, but are needed to run the application. Use this option if you want to apply restrictions to those indirect dependencies.

```
rules:
  - selectorRegexp: k8s[.]io
    allowedPrefixes:
      - k8s.io/gengo/examples
      - k8s.io/kubernetes/third_party
    forbiddenPrefixes:
      - k8s.io/kubernetes/pkg/third_party/deprecated
  - selectorRegexp: ^unsafe$
    forbiddenPrefixes:
      - ""
inverseRules:
  - selectorRegexp: k8s[.]io
    allowedPrefixes:
      - k8s.io/same-repo
      - k8s.io/kubernetes/pkg/legacy
    forbiddenPrefixes:
      - k8s.io/kubernetes/pkg/legacy/subpkg
  - selectorRegexp: k8s[.]io
    transitive: true
    forbiddenPrefixes:
      - k8s.io/kubernetes/cmd/kubelet
      - k8s.io/kubernetes/cmd/kubectl
```

How do I run import-boss within the k/k repo?

- In order to include _test.go files, make sure to pass in the `include-test-files` flag:

```
hack/verify-import-boss.sh --include-test-files=true
```

- To include other directories, pass in a directory or directories using the `input-dirs` flag:

```
hack/verify-import-boss.sh --input-
dirs="k8s.io/kubernetes/test/e2e/framework/..."
```

Reference

- import-boss