Translations README

This is a basic sketch of the workflow needed to add translations:

Adding/Updating Translations

New languages

Create `staging/src/k8s.io/kubectl/pkg/util/i18n/translations/kubectl/<language>/LC_MESSAGES/k`
There's no need to update `translations/test/...` which is only used for unit tests.

There is an example PR here which adds support for French.

Once you've added a new language, you'll need to register it in `staging/src/k8s.io/kubectl/pkg/util/i18n/i` by adding it to the `knownTranslations` map.

Wrapping strings

There is a simple script in `staging/src/k8s.io/kubectl/pkg/util/i18n/translations/extract.py` that performs simple regular expression based wrapping of strings. It can always use improvements to understand additional strings.

Extracting strings

Once the strings are wrapped, you can extract strings from go files using the `go-xgettext` command which can be installed with:

```
go get github.com/gosexy/gettext/go-xgettext
```

Once that's installed you can run `./hack/update-translations.sh`, which will extract and sort any new strings.

Adding new translations

Edit the appropriate `k8s.po` file, `poedit` is a popular open source tool for translations. You can load the `staging/src/k8s.io/kubectl/pkg/util/i18n/translations/kubectl/template.po` file to find messages that might be missing.

Once you are done with your `k8s.po` file, generate the corresponding `k8s.mo` file. `poedit` does this automatically on save, but you can also run `./hack/update-translations.sh` to perform the `po` to `mo` translation.

We use the English translation as the `msgid`.

Regenerating the bindata file

> Note: Regeneration of bindata is no more necessary for Kubernetes 1.22+ as the translations are now embedded into the binary at compile

time. See: https://github.com/kubernetes/kubernetes/pull/99829

With the `mo` files up to date, you can now convert the generated files into code using `go-bindata` command which can be installed with:

`go get github.com/go-bindata/go-bindata/...`

Run `./hack/generate-bindata.sh`, this will turn the translation files into generated code which will in turn be packaged into the Kubernetes binaries.

Extracting strings

There is a script in `staging/src/k8s.io/kubectl/pkg/util/i18n/translations/extract.py` that knows how to do some simple extraction. It needs a lot of work.

Using translations

To use translations, you simply need to add:

```
import pkg/i18n
...
// Get a translated string
translated := i18n.T("Your message in english here")

// Get a translated plural string
translated := i18n.T("You had % items", items)

// Translated error
return i18n.Error("Something bad happened")

// Translated plural error
return i18n.Error("%d bad things happened")
```