API serialization compatibility tests

This directory tree contains serialized API objects in json, yaml, and protobuf formats.

Populating data for each release

After every v1.x.0 release, snapshot compatibility data.

For example, to capture compatibility data for `v1.20.0`:

```
export VERSION=v1.20.0
git checkout ${VERSION}
cp -fr staging/src/k8s.io/api/testdata/{HEAD,${VERSION}}
git checkout -b ${VERSION}-api-testdata master
git add .
git commit -m "Add ${VERSION} API testdata"
```

Current version

The `HEAD` subdirectory contains serialized API objects generated from the current commit:

```
HEAD/
  <group>.<version>.<kind>.[json|yaml|pb]
```

To run serialization tests just for the current version:

```
go test k8s.io/api -run //HEAD
```

All three formats of a given group/version/kind are expected to decode successfully to identical objects, and to round-trip back to serialized form with identical bytes. Adding new fields or API types *is* expected to modify these fixtures. To regenerate them, run:

```
UPDATE_COMPATIBILITY_FIXTURE_DATA=true go test k8s.io/api -run //HEAD
```

Previous versions

The vX.Y.Z subdirectories contain serialized API objects from previous releases:

```
vX.Y.Z/
  <group>.<version>.<kind>.[json|yaml|pb]
```

All three formats of a given group/version/kind are expected to decode successfully to identical objects, and to round-trip back to serialized form with identical bytes. Adding new fields to existing API types is *not* expected to require modifications to these fixtures. This requires making optional scalar and struct fields pointers so that protobuf serialization omits them when not present.

To run serialization tests just for a previous version, like `v1.14.0`:

```
go test k8s.io/api -run //v1.14.0
```

To run serialization tests for a particular group/version/kind, like `apps/v1` `Deployment`:

```
go test k8s.io/api -run /apps.v1.Deployment/
```

Failures to decode, to round-trip identical bytes, or to decode identical objects from json/yaml/protobuf, will output detailed errors about the differences encountered. Detailed errors about protobuf differences requires `protoc` to be available on your `$PATH`.

In exceptional cases, new non-pointer fields were added to existing API types that serialized zero values, resulting in additional fields being output when round-tripping data from previous releases, and failing round-trip tests.

To resolve this, a `<group>.<version>.<kind>_after_roundtrip.[json|yaml|pb]` file containing the expected data after roundtripping can be placed beside the serialized data file from a previous release.

These `after_roundtrip` files are generated by running the failing round-trip tests with `UPDATE_COMPATIBILITY_FIXTURE_DATA=true` set. The detailed diff from the test failure should be included in the commit message, along with a reference to the change that caused the failure. Updates to these files is exceptional, and requires extremely close review to ensure we are not breaking backwards compatibility with serialized data from previous releases.

To see the diff between the original JSON/YAML data and the `...after_roundtrip...` files:

```
cd vendor/k8s.io/api/testdata/v1.14.0/
diff -u admission.k8s.io.v1beta1.AdmissionReview.json admission.k8s.io.v1beta1.AdmissionReview.after_roundtrip.json
diff -u admission.k8s.io.v1beta1.AdmissionReview.yaml admission.k8s.io.v1beta1.AdmissionReview.after_roundtrip.yaml
```

```
--- admission.k8s.io.v1beta1.AdmissionReview.json	2019-06-02 20:21:03.000000000 -0400
+++ admission.k8s.io.v1beta1.AdmissionReview.after_roundtrip.json	2019-06-02 20:21:03.000000000 -0400
@@ -31,7 +31,8 @@
     },
     "object": {"apiVersion":"example.com/v1","kind":"CustomType","spec":{"replicas":1},"status":{"available":1}},
     "oldObject": {"apiVersion":"example.com/v1","kind":"CustomType","spec":{"replicas":1},"status":{"available":1}},
-    "dryRun": true
+    "dryRun": true,
+    "options": null
   },
   "response": {
     "uid": "爗¼ªov鉿",
```

```
--- admission.k8s.io.v1beta1.AdmissionReview.yaml	2019-06-02 20:21:03.000000000 -0400
+++ admission.k8s.io.v1beta1.AdmissionReview.after_roundtrip.yaml	2019-06-
```

```
02 20:21:03.000000000 -0400
@@ -23,6 +23,7 @@
     status:
       available: 1
   operation: 祈¡ıŵDz廮ê{sŊəp饕姥咨鍚
+  options: null
   resource:
     group: "5"
     resource: "7"
```

To see the diff between the original proto data and the `...after_roundtrip...` file, you must have `protoc` available, and strip off the leading four-byte kubernetes protobuf header to get standard protobuf that can be decoded:

```
cd vendor/k8s.io/api/testdata/v1.14.0/
diff -u \
  <(tail -c +5 admission.k8s.io.v1beta1.AdmissionReview.pb | protoc --decode_raw) \
  <(tail -c +5 admission.k8s.io.v1beta1.AdmissionReview.after_roundtrip.pb | protoc
--decode_raw)
```

```
--- /dev/fd/63  2019-06-03 11:56:12.000000000 -0400
+++ /dev/fd/62  2019-06-03 11:56:12.000000000 -0400
@@ -37,6 +37,8 @@
       1: "{\"apiVersion\":\"example.com/v1\",\"kind\":\"CustomType\",\"spec\":
{\"replicas\":1},\"status\":{\"available\":1}}"
     }
     11: 1
+    12: ""
+    15: ""
   }
   2 {
     1: "\347\210\237\302\274\302\252ov\351\210\266"
```