Generate OpenAPI definitions

- To generate definition for a specific type or package add "+k8s:openapi-gen=true" tag to the type/package comment lines.
- To exclude a type or a member from a tagged package/type, add "+k8s:openapi-gen=false" tag to the comment lines.

OpenAPI Extensions

OpenAPI spec can have extensions on types. To define one or more extensions on a type or its member add `+k8s:openapi-gen=x-kubernetes-$NAME:$VALUE` to the comment lines before type/member. A type/member can have multiple extensions. The rest of the line in the comment will be used as $VALUE so there is no need to escape or quote the value string. Extensions can be used to pass more information to client generators or documentation generators. For example a type might have a friendly name to be displayed in documentation or being used in a client's fluent interface.

Custom OpenAPI type definitions

Custom types which otherwise don't map directly to OpenAPI can override their OpenAPI definition by implementing a function named "OpenAPIDefinition" with the following signature:

```go
import openapi "k8s.io/kube-openapi/pkg/common"

// ...

type Time struct {
    time.Time
}

func (_ Time) OpenAPIDefinition() openapi.OpenAPIDefinition {
    return openapi.OpenAPIDefinition{
        Schema: spec.Schema{
            SchemaProps: spec.SchemaProps{
                Type:   []string{"string"},
                Format: "date-time",
            },
        },
    }
}
```

Alternatively, the type can avoid the "openapi" import by defining the following methods. The following example produces the same OpenAPI definition as the example above:

```
func (_ Time) OpenAPISchemaType() []string { return []string{"string"} }
func (_ Time) OpenAPISchemaFormat() string { return "date-time" }
```