Generating SDL

> warning **Warning** This chapter applies only to the code first approach.

To manually generate a GraphQL SDL schema (i.e., without running an application, connecting to the database, hooking up resolvers, etc.), use the `GraphQLSchemaBuilderModule`.

```
async function generateSchema() {
  const app = await NestFactory.create(GraphQLSchemaBuilderModule);
  await app.init();

  const gqlSchemaFactory = app.get(GraphQLSchemaFactory);
  const schema = await gqlSchemaFactory.create([RecipesResolver]);
  console.log(printSchema(schema));
}
```

> info **Hint** The `GraphQLSchemaBuilderModule` and `GraphQLSchemaFactory` are imported from the `@nestjs/graphql` package. The `printSchema` function is imported from the `graphql` package.

Usage

The `gqlSchemaFactory.create()` method takes an array of resolver class references. For example:

```
const schema = await gqlSchemaFactory.create([
  RecipesResolver,
  AuthorsResolver,
  PostsResolvers,
]);
```

It also takes a second optional argument with an array of scalar classes:

```
const schema = await gqlSchemaFactory.create(
  [RecipesResolver, AuthorsResolver, PostsResolvers],
  [DurationScalar, DateScalar],
);
```

Lastly, you can pass an options object:

```
const schema = await gqlSchemaFactory.create([RecipesResolver], {
  skipCheck: true,
  orphanedTypes: [],
});
```

- `skipCheck`: ignore schema validation; boolean, defaults to `false`
- `orphanedTypes`: list of classes that are not explicitly referenced (not part of the object graph) to be generated. Normally, if a class is declared but isn't otherwise referenced in the graph, it's omitted. The property value is an array of class references.