Asynchronous providers

At times, the application start should be delayed until one or more **asynchronous tasks** are completed. For example, you may not want to start accepting requests until the connection with the database has been established. You can achieve this using asynchronous providers.

The syntax for this is to use `async/await` with the `useFactory` syntax. The factory returns a `Promise`, and the factory function can `await` asynchronous tasks. Nest will await resolution of the promise before instantiating any class that depends on (injects) such a provider.

```
{
  provide: 'ASYNC_CONNECTION',
  useFactory: async () => {
    const connection = await createConnection(options);
    return connection;
  },
}
```

> info **Hint** Learn more about custom provider syntax here.

Injection

Asynchronous providers are injected to other components by their tokens, like any other provider. In the example above, you would use the construct `@Inject('ASYNC_CONNECTION')`.

Example

The TypeORM recipe has a more substantial example of an asynchronous provider.