

Quarkus - Kubernetes Config

Quarkus includes the `kubernetes-config` extension which allows developers to use Kubernetes `ConfigMaps` as a configuration source, without having to mount the `ConfigMaps` into the `Pod` running the Quarkus application.

Configuration

Once you have your Quarkus project configured you can add the `kubernetes-config` extension by running the following command in your project base directory.

```
./mvnw quarkus:add-extension -Dextensions="kubernetes-config"
```

This will add the following to your `pom.xml`:

```
<dependency>
  <groupId>io.quarkus</groupId>
  <artifactId>quarkus-kubernetes-config</artifactId>
</dependency>
```

Usage

The extension works by reading `ConfigMaps` directly from the Kubernetes API server using the `Kubernetes Client`.

The extension understands the following types of `ConfigMaps` as input sources:

- `ConfigMaps` that contain literal data (see [this](#) for an example on how to create one)
- `ConfigMaps` created from files named, `application.properties`, `application.yaml` or `application.yml` (see [this](#) for an example on how to create one).

To configure which `ConfigMaps` (from the namespace that the application runs in, or the explicitly configured namespace via `quarkus.kubernetes-client.namespace`), you can set the `quarkus.kubernetes-config.config-maps` property (in any of the usual Quarkus ways). Keep in mind however that you will also have to explicitly enable the retrieval of `ConfigMaps` by setting `quarkus.kubernetes-config.enabled=true` (the default is `false` in order to make it easy to test the application locally).

Priority of obtained properties


The properties obtained from the `ConfigMaps` have a higher priority (i.e. they override) any properties of the same name that are found in `application.properties` (or the YAML equivalents), but they have lower priority than properties set via Environment Variables or Java System Properties.

Kubernetes Permissions

Since reading ConfigMaps involves interacting with the Kubernetes API Server, when [RBAC](#) is enabled on the cluster, the [ServiceAccount](#) that is used to run the application needs to have the proper permissions for such access.

Thankfully, when using the `kubernetes-config` extension along with the [Kubernetes](#) extension, all the necessary Kubernetes resources to make that happen are automatically generated.

Configuration Reference

 Configuration property fixed at build time - All other configuration properties are overridable at runtime

| Configuration property | Type | Default |
|---|----------------|--------------------|
| <code>quarkus.kubernetes-config.enabled</code> If set to true, the application will attempt to look up the configuration from the API server | boolean | <code>false</code> |
| <code>quarkus.kubernetes-config.fail-on-missing-config</code> If set to true, the application will not start if any of the configured config sources cannot be located | boolean | <code>true</code> |
| <code>quarkus.kubernetes-config.config-maps</code> ConfigMaps to look for in the namespace that the Kubernetes Client has been configured for | list of string | |