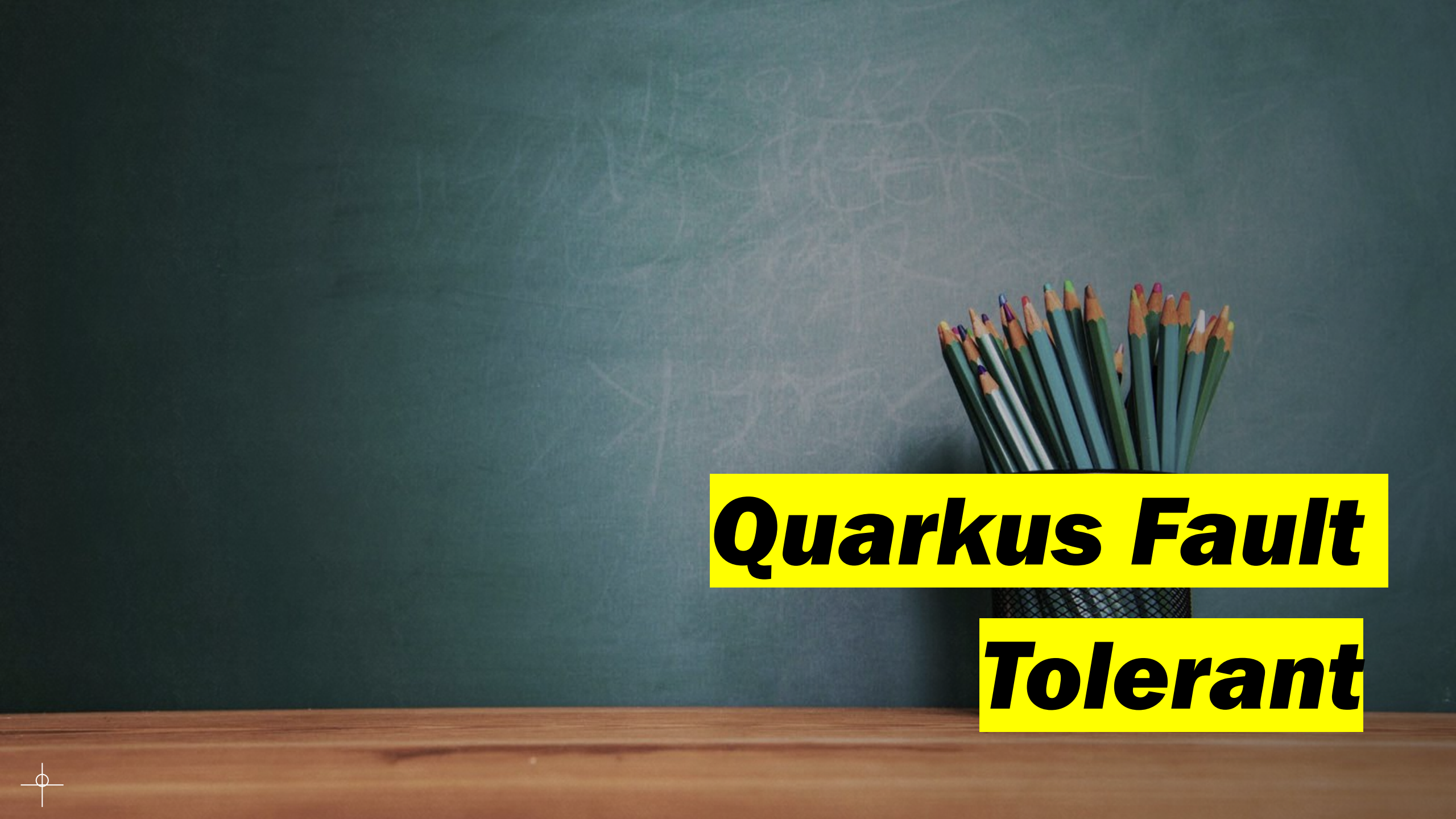
Quarkus Fault
Tolerant

Retry Policy (`@org.eclipse.microprofile.faulttolerance.Retry`)
Defines the conditions for retrying operations.

Timeout (`@org.eclipse.microprofile.faulttolerance.Timeout`)
Defines the maximum execution time before interrupting a request.

Fallback (`@org.eclipse.microprofile.faulttolerance.Fallback`)
Executes an alternative method when the primary method fails.

Circuit Breaker (`@org.eclipse.microprofile.faulttolerance.CircuitBreaker`)
Activates a fail-fast mechanism if the system is overloaded or unavailable.

SmallRye Fault Tolerance library

```
[user@host myapp]$ mvn quarkus:add-extension \
-Dextensions=smallrye-fault-tolerance
```

1) Retry

```
@Retry( maxRetries = 5, retryOn = RuntimeException.class )
public Product maybeFail(int id) {
  ...implementation omitted...
}
```

2) Timeout

```
@Timeout( 3000 )
public Product getProduct( int id ) {
    ...implementation omitted...
}
```

Typically, you should handle the `TimeoutException` error, either with an explicit `try-catch` block or by using other fault tolerance annotations, to provide a graceful response.

```
try {
   productService.getProduct( productId );
} catch (TimeoutException e) {
   ...implementation omitted...
}
```

3) Fallback

```
@Fallback( fallbackMethod = "getCachedProduct" )
public Product getProduct( int id ) {
    ...implementation omitted...
}

public Product getCachedProduct( int id ) {
   ...implementation omitted...
}
```

3) Fallback – Alternativa Clase

```
public class ProductResource {

   @Fallback( ProductFallback.class )
   public String getProduct(int id) {
      ...implementation omitted...
   }
}

public class ProductFallback {

   public String handle( ExecutionContext context ) {
      ...implementation omitted...
   }
}
```

```
public class ProductCacheFallback implements FallbackHandler<Product> {
   ...implementation omitted...

   @Override
   public Product handle(ExecutionContext executionContext) {
       int id = executionContext.getParameters()[0];
       return cache.getProductForId(id);
   }
}
```

Circuit Breaker

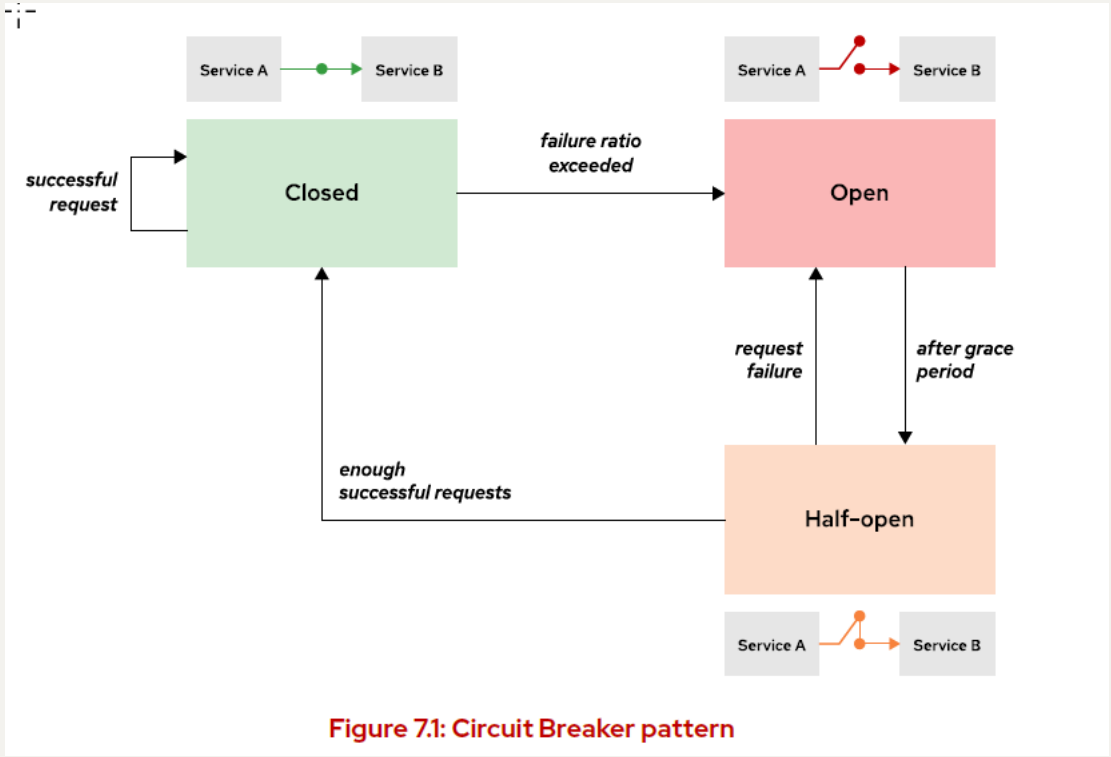

Figure 7.1: Circuit Breaker pattern

```
@CircuitBreaker(
   requestVolumeThreshold = 4, (1)
   failureRatio = 0.5, (2)
   delay = 1000 (3)
)
public List<String> getProducts() {
   ...implementation omitted...
}
```

```
@CircuitBreaker( requestVolumeThreshold = 4 )
@Fallback( fallbackMethod = "recommendMostPopularProducts" )
public List<String> recommendProductsByUserPreference(User user) {
   ...implementation omitted...
}
```

Integrando con Microprofile Config

```
<classname>/<methodname>/<annotation>/<parameter>
```

For example, the following line is equivalent to adding the `@Retry(maxRetries=5)` annotation to the `getProduct` method of the `com.acme.ProductResource` class.

```
com.acme.ProductResource/getProduct/Retry/maxRetries=5
```

Integrando con Microprofile Metrics

quarkus-smallrye-metrics

For example, the `@Fallback` annotation generates a `ft.<name>.fallback.calls.total` metric, where `<name>` is the fully qualified name of the original method. This metric counts the number of times a fallback method is used.