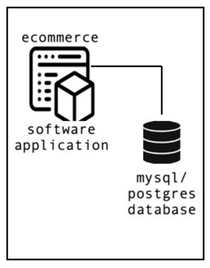


The software applications may have to communicate with external resources or systems inorder to fulfill the business functionality like
1. database server
2. messaging server
3. caching server
4. file server
etc
to perform various different operations.

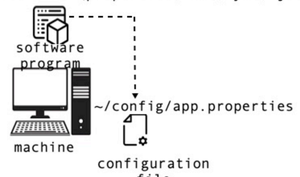
The developer has to write relevant programming logic within the program/application inorder to communicate with the external resource or the system. In the application program he has to refer to the details of the external resource to whom the application has to talk to like
1. address or remoteResourceURL
2. credentails (username/password)
and any other configuration information based on the remote resource/system we are connecting with to perform the operation.

How can the developer can populate the information about the external resources/systems within the Program?
There are several ways of accomplishing it
#1. The developer can write directly the configuration information pertaining to the external resources aspart of the program itself. We can refer the below code snippet to understand:

```
// here in the java program the developer is writing the code to connect to the database server
Connection con = DriverManager.getConnection("jdbc:mysql://127.0.0.1:3306/databasename", "root", "welcome1");
                                              [dbhost]:[dbport]             [dbuser] [dbpassword]
```

But this approach has several dis-advantages even though it is easy to accomplish
1. if there is a change in the configuration information of the external resource, the developer has to modify the sourcecode of the program, rebuild, repackage, redeploy and restart the server/program inorder to get the changes effected, it incurrs huge amount of efforts in making and rolling these configuration changes.
2. our application moves across different stages of delivery like test, stage, uat, prod etc. The configuration information pertaining to these external resources differs for each environment. So every time inorder to move the application across these envs, the developers has to modify the code, rebuild, repackage and deploy it which seems to be back to the original problem
From this we can understand writing the configuration information directly inside the program doesnt seem to be a good approach.

#2 Instead of writing the configuration information about the external resources aspart of the code, we can externalize the configuration in configuration file like (properties, xml, yml, json etc) and read that information within the program using which perform the operations on the external resources.



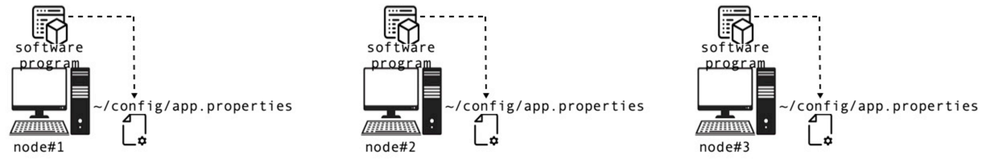

```
app.properties
---------------
db.host=127.0.0.1
db.port=3306
db.dbname=roadsterdb
db.username=root
db.password=welcome1
```

```
program
--------
// read the properties file from the ${user.home}/config/app.properties file
Properties props = new Properties();
props.load(new FileInputStream(new File("${user.home}/config/app.properties"));

Connection con = DriverManager.getConnection("jdbc:mysql://"+props.getProperty("db.host")
+":"+props.getProperty("db.port")+"/"+props.getProperty("db.dbname"),
props.getProperty("db.username"), props.getProperty("db.password"));
```

If we refer the above block of code, the program is reading the configuration from an external file and using that information it is connecting or communicating with the external resource/system. So any change in the external resource/ system, we only need to modify the relevant configuration in the configuration file and restart the program. We dont need to modify the sourcecode of the program, rebuild, repackage, redeploy and restart hence it saves lot of time and efforts in rolling the application across different environments.

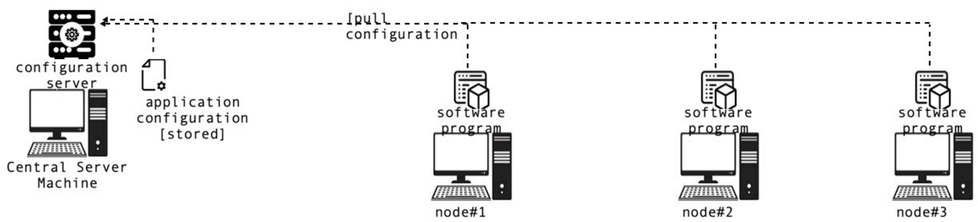
Looks like this approach has solved most of the problems we discussed earlier, but it has lot of challenges as well.
challenges or problems:
In production environment, the application will be deployed on multiple nodes of the cluster for high availability and scalability. In such case the configuration information would be distributed across all the nodes of the cluster where our application is running



when we run our application in a cluster environment, we create multiple copies of the configuration information per each node on which our application is deployed, which leads to maintainance problem. when there is a change in external resource like an database server password change, then we need to goto each node on the cluster and modify the configuration file and restart the application which is an tedious job

---

#3. centralize the configuration and distribute across all the instances of the application
Instead of creating multiple copies of the configuration per each node of the cluster, we can place the application configuration in an central server location and distribute to all the nodes on which our application is running.



When there is a change in the external resource, modify the configuration in ConfigurationServer (only in 1 place) and restart the programs that are running on the nodes of the cluster, so that the programs will pull latest copy of the configuration and will run quickly with those changes.

even though this seems to solve all the problems discussed earlier, still it has few problems of its own:
Limitations:
1. our application is coded to read the configuration from the ConfigServer location, if there is a change in the configuration server from which our application has to read again we need to modify the application logic
2. our application is designed to work with a specific configuration file format for eg.. YAML, and in future we want to switch to JSON again we need to modify the code inside the application.

From the above we can understand any of these approaches of reading/using the configuration in our application has their own limitation. If we look at them we are running into these problems if our application is reading the configuration from external source. Instead if the software developer can design the application allowing to pass the configuration as an input, then the application will not have these problems

```
String host = System.getProperty("db.host");
String port = System.getProperty("db.port");
String username = System.getProperty("db.username");
String password = System.getProperty("db.password");
Connection con = DriverManager.getConnection("jdbc:mysql://"+host+":"+port+"/dbname", username, password);
```

In the above code, the program is not reading the configuration from
1. a specific configuration file
2. from a specific config server location
3. a specific configuration file format

it is reading the configuration values from the env or host on which the application is running. So our application is agnostic to a specific file/location/ format completely.
So the user while running the application, should be able to pass the configuration by populating on that env, our application will work.

How to populate the configuration information onto the env while launching or running the application, so that our application can use that configuration while executing?
So if we are running our application as an docker container, then while lauching the application on that container, we can pass env variables using --env-file