**QCC2CPLAT-1084 - As a C2C admin, want to host application configuration files for multi region and multi environment hosting.**

**QCC2CPLAT-1181 - Analyse the Config server with S3 and add up the pros and cons with a POC**

**PROS**

1. Fast and easy to read.
2. Easy to modify, if we have the access to S3.
3. Versioning is possible.
4. Not publicly accessible. (Choice is there to make Private and provide Read/Write accessibility).
5. Regional support for storing assets in particular locations around the world.
6. Notifications on Object changes – Notification can be provided if some S3 operations are performed on the file.
7. Per-file permission system (make a file public or not).
8. S3’s “Standard” storage class offers very high durability (it advertises 99.999999999% durability, or “eleven 9s”), high availability, low latency access, and relatively cheap access cost.
9. Easy to change to other data-source/backend.

**CONS**

Need to change the bootstrap.properties when u want to deploy in different regions...or we can also pass as argument to the command by setting the property.

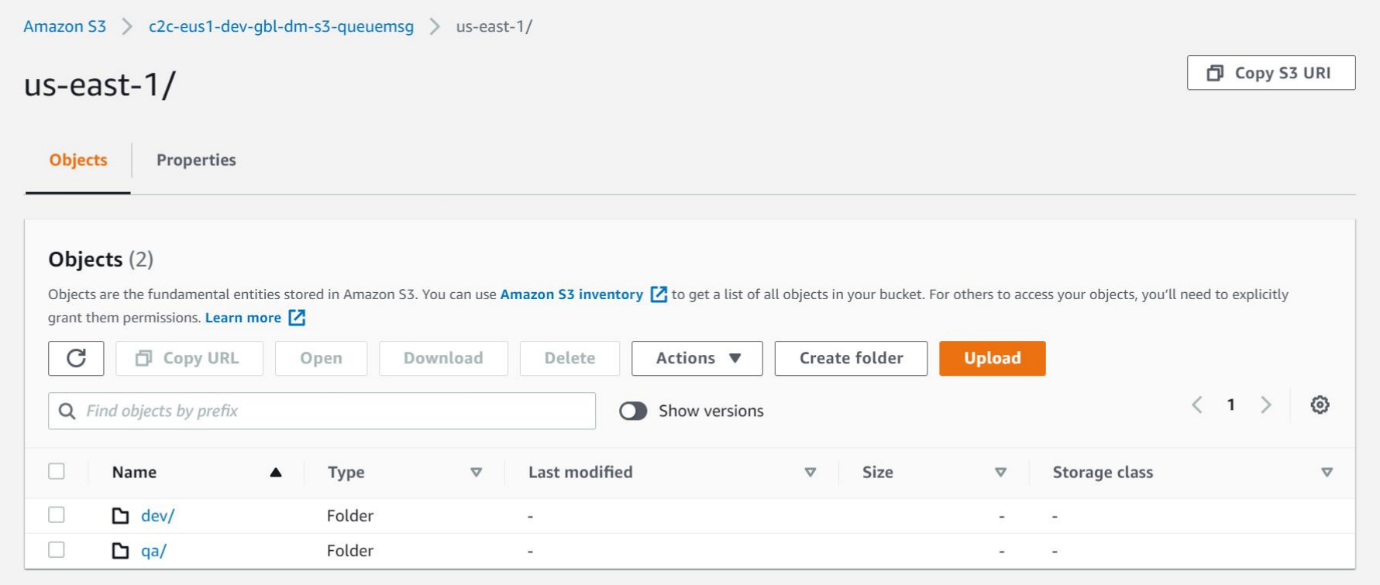
**Approach we are planning to implement in case of S3**

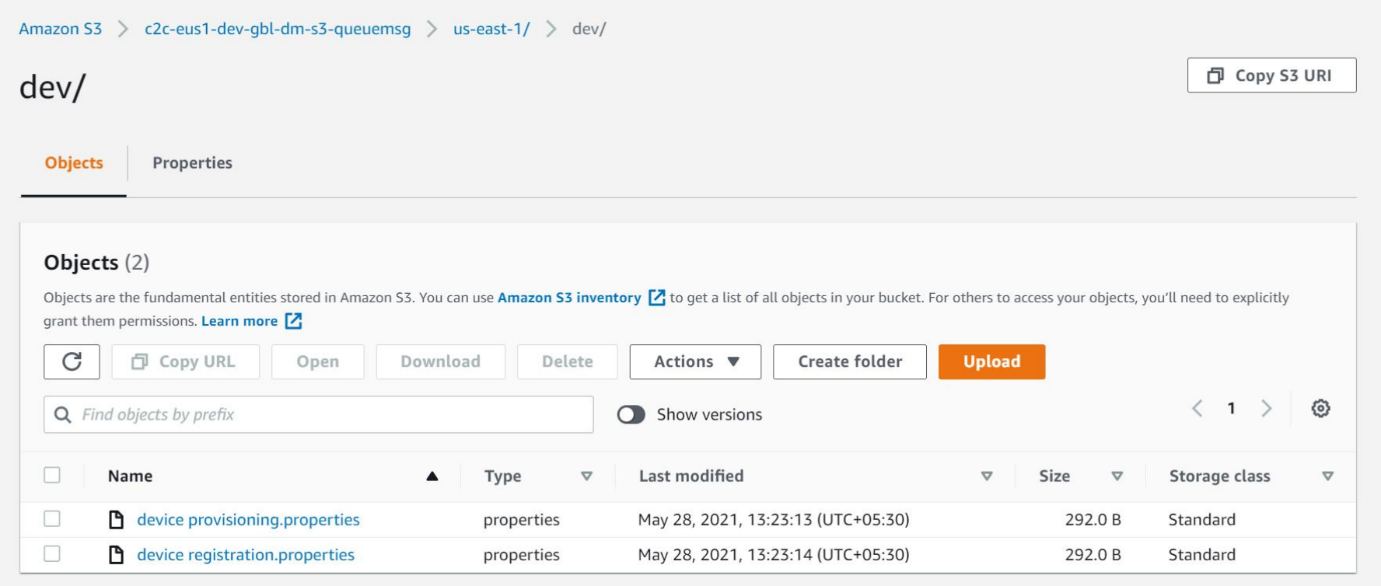
If we are following the folder structure for region and the environment then we only require to provide the following properties in the bootstrap.properties of config client application.

**spring.application.name=device registration**

**spring.cloud.config.label=us-east-1/dev**

us-east-1/dev specifies the file path/ key (as folder) in the bucket. So, we can maintain each region and its own environment specific property files in folder structure as represented in the below screenshots.





**Note**: Tested the latency scenario using S3, by keeping the config server in us-east-2 and config client service in Mumbai region…We have found a latency of 1 milli second.

**QCC2CPLAT-1182 - Analyse the config server with Database and add up the pros and cons with POC**

**PROS**

1. Management - You can create an interface that your users can access to change settings. You can easily programmatically change config values through an admin interface when needed, which can enforce logic around what can go into each setting. You can't do that so easily with a file (though, of course, it is possible).

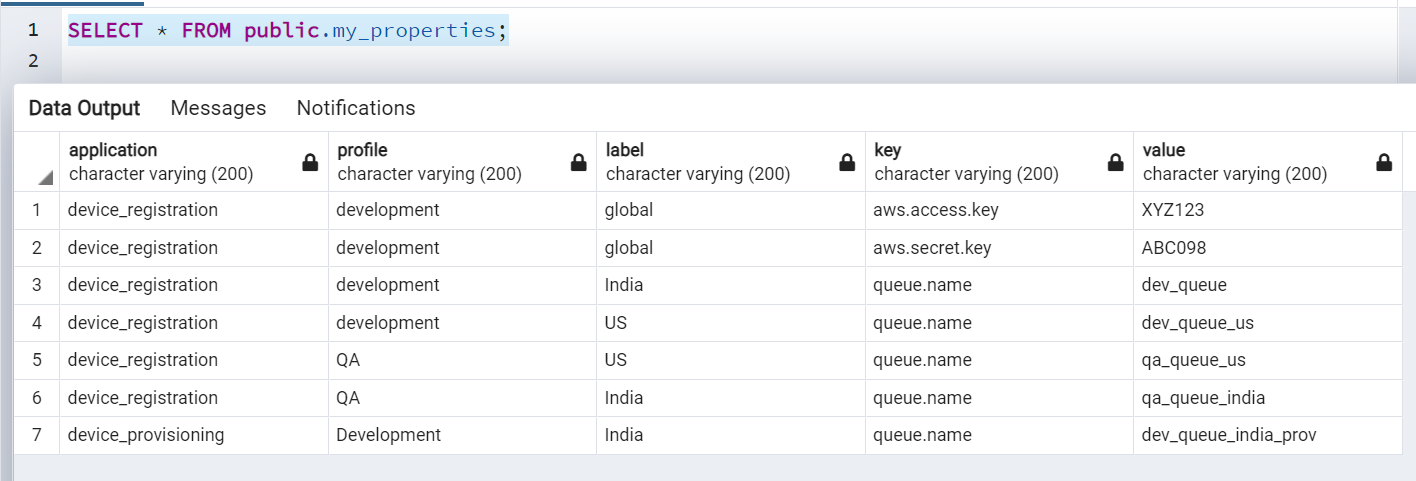
**CONS**

1. Relies on database connection.
2. Overhead when reading from database.
3. If you don't provide a tool (UI), it might be more difficult for sys admins to administer the product rather than a file. (If there is no proper prerequisite knowledge about the SQL commands or a simple mistake of not adding ‘WHERE clause’ can affect the entire property/system)
4. If you migrate a database to a different environment (e.g. a production database to test to do some issue checking) your settings are included.
5. Damage to database affects virtually all application programs. Need extra efforts to have reliability and durability.
6. Maintain versioning is an extra activity here
7. Difficult to change to other data-source/backend.

**Approach we are planning to implement in case of Database**

We will be creating a database and a table to store the properties. In the below screenshot of the table, as you can see we have the following columns.

* **Application**: To specify to which application that property is specified to.
* **Profile**: Specified to which environment it is specific to.
* **Label**: Can be used to specify the Region to which the property exists.
* **Key** : Key of that property
* **Value** : Value associated with that specific key



**QCC2CPLAT-1183 - Analyse the issues with the existing AWS code commit for the config server**

**PROS**

1. Unlimited Scalability- AWS Code Commit repositories can scale up to meet your development needs. The service can handle repositories with large numbers of ﬁles or branches, large ﬁle sizes, and lengthy revision histories. AWS Code Commit has no limit on the size of your repositories and on the stored ﬁle types.
2. Integration with Other AWS Development Services - Code Commit provides seamless integration with Code Build, Code Star, Lambda, Cloud9, Code Pipeline, Cloud Formation, SNS, and other AWS service and products that make software development, build, deployment and release extremely fast.
3. Secure way to access
4. Easy to change to other data-source/backend.
5. Not publicly accessible

**CONS**

1. Need to check the feasibility to access using https

**Notes**:

* Depend upon the application usage/requirement we have Hot refresh (using the /refresh actuator) and cold refresh (with the normal redeployment of the service after property changes).
* Most of the real-time, it is preferred and advisable to use the cold refresh, which has some advantages like not require to have the PROD change request.
* Git Refresh Rate - You can control how often the config server will fetch updated configuration data from your Git backend by using **spring.cloud.config.server.git.refreshRate**. The value of this property is specified in seconds. By default the value is 0, meaning the config server will fetch updated configuration from the GIT repo every time it is requested.
* You can configure the time, in seconds, that the configuration server will wait to acquire an HTTP connection.

spring.cloud.config.server.git.timeout=4

**Conclusions:**

* Verified few other Architect team about the real-time usage of config-server in PROD ENV, heard mixed response of using the GIT related(like bit bucket/aws code-commit) and database usage for Config server.
* It depends on the Client requirement/adaption and the application usage.
* I could see weightage in all the options, for database we require additional maintenances like versioning, access provisioning.
* Either the AWS code-commit or S3 are more feasible to options.

**Reference Links:**

<https://aws.amazon.com/getting-started/hands-on/aurora-global-database/>

<https://docs.spring.io/spring-cloud-config/docs/current/reference/html/#_aws_s3_backend>

<https://medium.com/swlh/spring-cloud-config-server-composite-configuration-jdbc-redis-awss3-d849c4d94383>

<https://stackoverflow.com/questions/1841365/is-it-better-to-store-platform-configuration-in-database-or-a-file#:~:text=First%2C%20you%20create%20a%20database,can%20access%20to%20change%20settings>.

<https://tanzu.vmware.com/developer/guides/spring/spring-cloud-config-set-up/>