Distributed KV Store (MUSIC sub-project)

Project Name: Distributed KV Store for configuration settings

Project description:

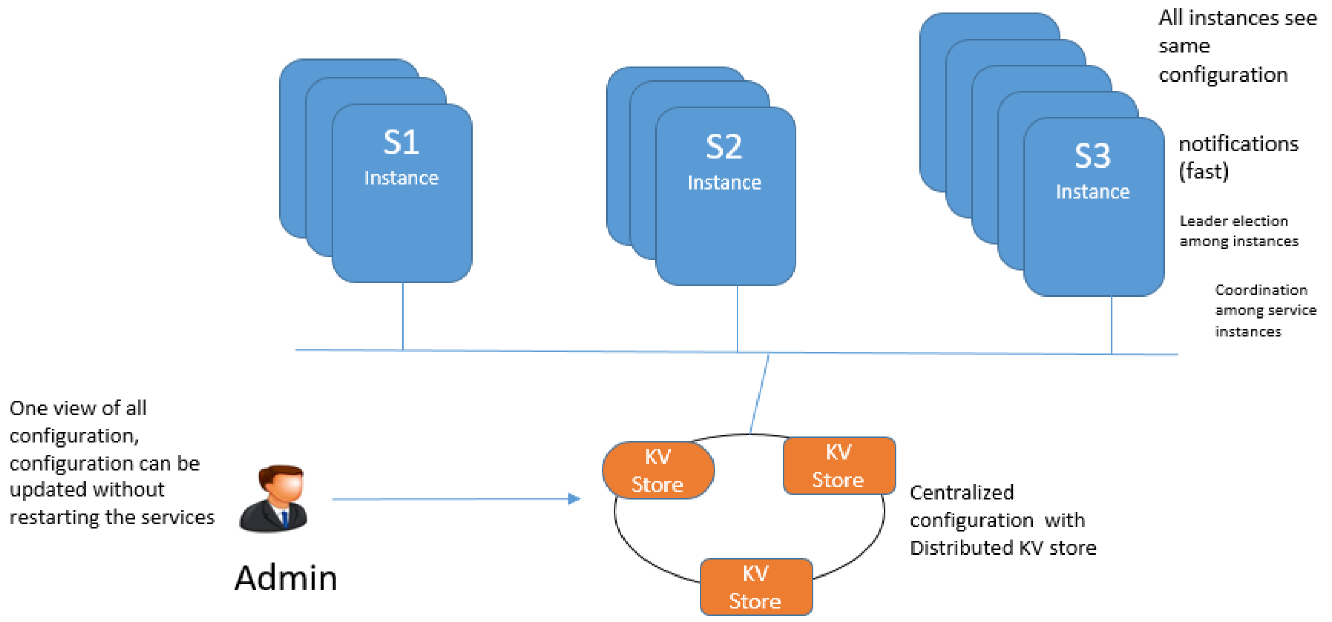
Requirements:

- Currently, the configuration properties are stored in files. Each ONAP project maintains its settings in its own configuration files. Some of the challenges & requirements are :
  - Any update to the configuration settings require restart of the service. This could lead to ONAP service unavailability for some time. Requirement is that any configuration update should not lead to restart of ONAP services.
  - There is good amount of duplication information among the files in various ONAP projects. Due to this, any change in this common information requires updating multiple files, which is error prone.
  - In future releases, ONAP is going to bring up services with multiple instances for high availability. Any change in the configuration should be reflected in all service instances and also the configuration should be consistent in all instances.
  - Configuration store itself should be distributed for high availability of configuration data.
  - Configuration change should be easier for administrator - GUI facility to update the configuration.
- When there is a cluster of service instances, there can be a need for one of the instances to become a leader. That is, leader election could be a requirement for services.
- Yet times, some operations need to be atomically performed and hence there could be a requirement for distributed lock.

Distributed KV Store project is expected to address above requirements. It is expected to provide following functionality

- Centralization of configuration
- Common configuration settings without duplication
- Configuration update without the need for restarting services
- GUI/CLI to update the configuration settings
- Auto notification of services when there is a configuration change.
- Reliable and distributed data storage.
- Leadership selection
- Distributed lock

Pictorial representation of Distributed KV Store with consuming services:


All instances see same configuration
notifications (fast)
Leader election among instances
Coordination among service instances
S1 Instance
S2 Instance
S3 Instance
One view of all configuration, configuration can be updated without restarting the services
Admin
KV Store
KV Store
KV Store
Centralized configuration with Distributed KV store

Scope:

Functionality :

- Ability to store configuration settings using reliable and distributed KV storage systems.
- Ability to provide leadership election among micro services.
- Ability to provide semaphore across service instances of ONAP services.
- Ability to modify the configuration settings
- Ability to inform active ONAP services of configuration changes.
- Ability to read the changed configuration
- config-seed micro service to load default values of configuration settings, if settings are not present in the distributed KV storage.
- Bootstrap phase functionality where each service upon startup reads the configuration settings from distribtued KV storage.

Interface/API:

- RESTful API to set/read/update the configuration settings, leadership election and take/release distributed lock.
- Long poll for near instant configuration notifications.
- Client packages:
    - For configuration store, both Java and Python have abstraction alrady over consul and hence there is no special service-level package is required.
    - Python and Java based client package to simplify leadership election and distirbuted locks. This package wraps various operations required to elect the leader. Also, this client package abstracts various operations required to create lock, take and release the lock.

Architecture Alignment:

- How does this project fit into the rest of the ONAP Architecture?
    - Configuration-seed micro service is expcted to be a common service.
    - consul from MSB can be used as Distributed key store
    - Boot strap phase is expected to be part of each service
- What other ONAP projects does this project depend on?
    - It does not depend on any existing ONAP projects.
- Are there dependencies with other open source projects?
    - Consul (http://consul.io). Reasons for choosing consule:
        - Well maintained.
        - Used by many projects (e.g EdgexFoundry, vault project)
        - Good support for Java libraries (For Java based services, cfg4j is normally used to read the configuration. cfg4j has consul backend, python-envconsul package is useful for python based services)
        - It is alredy being used by ONAP (as part of MSB) as service-registry and service-discovery.

Other Information:

- Distributed KV Store : https://github.com/hashicorp/consul
- configuration-seed : https://github.com/edgexfoundry/core-config-seed (But suggest to implement this in golang, ideas can be taken from this project).

Key Project Facts:

Primary contact : Srinivasa Addepalli, Shashank Kumar Shankar

Facts	Info
PTL (first and last name)	
Jira Project Name	MUSIC
Jira Key	MUSIC
Project ID	org.onap.music.distributed-kv-store
Link to Wiki Space	Distributed KV Store (MUSIC sub-project)

Release Components Name:

Note: refer to existing project for details on how to fill out this table

Components Name	Components Repository name	Maven Group ID	Components Description
distributed-kv-store	music/distributed-kv-store	org.onap.music.distributed-kv-store	Source code to read initial configuration date into KV Store.

Resources committed to the Release:

Note 1: No more than 5 committers per project. Balance the committers list and avoid members representing only one company.

Note 2: It is critical to complete all the information requested, that will help to fast forward the onboarding process.

Role	First Name Last Name	Linux Foundation ID	Email Address	Location
PTL	Shashank Kumar Shankar	shashank.kumar.shankar	shashank.kumar.shankar@intel.com	Portland, OR
Committers	Shashank Kumar Shankar	shashank.kumar.shankar	shashank.kumar.shankar@intel.com	Portland, OR
Contributors				

Tracking Milestones:

https://wiki.onap.org/display/DW/Tracking+Milestones

API Reference:

https://git.onap.org/music/distributed-kv-store/tree/swagger.yaml swagger API.html

Testing and Integration:

The following type of tests are done in this release:

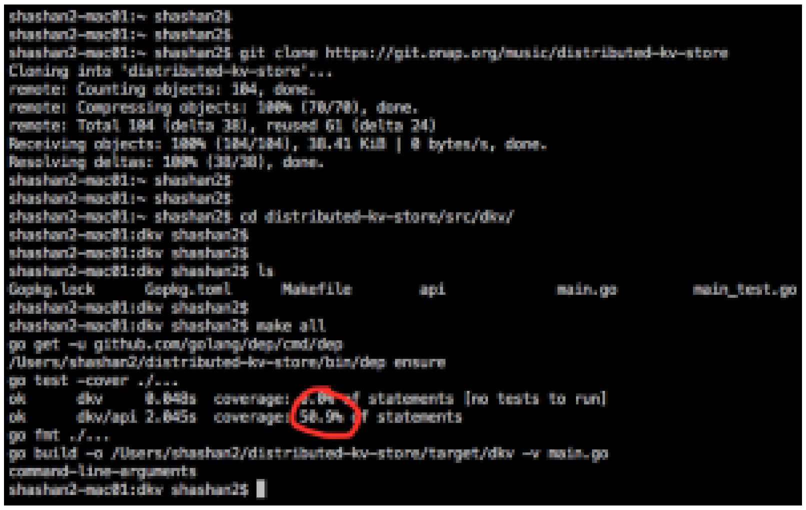
- Unit tests: Currently has around 50% unit test coverage as evidenced in the logs for the latest build: https://jenkins.onap.org/view/music/job/music-distributed-kv-store-master-merge-golang/lastBuild/consoleText
- Functional Test cases: Functional Tests run in CSIT tests.

Example test case: https://git.onap.org/music/distributed-kv-store/tree/src/dkv/api/configHandlers_test.go

Generate Code Coverage Report of source code for Golang applications:

```
$ go test -coverprofile=coverage.out
```

```
$ go tool cover -html=coverage.out
```



```
shashan2-mac01:~ shashan2$
shashan2-mac01:~ shashan2$
shashan2-mac01:~ shashan2$ git clone https://git.onap.org/music/distributed-kv-store
Cloning into 'distributed-kv-store'...
remote: Counting objects: 104, done.
remote: Compressing objects: 100% (70/70), done.
remote: Total 104 (delta 38), reused 61 (delta 24)
Receiving objects: 100% (104/104), 38.41 KiB | 0 bytes/s, done.
Resolving deltas: 100% (38/38), done.
shashan2-mac01:~ shashan2$
shashan2-mac01:~ shashan2$
shashan2-mac01:~ shashan2$ cd distributed-kv-store/src/dkv/
shashan2-mac01:dkv shashan2$
shashan2-mac01:dkv shashan2$
shashan2-mac01:dkv shashan2$ ls
Gopkg.lock       Gopkg.toml       Makefile        api             main.go         main_test.go
shashan2-mac01:dkv shashan2$
shashan2-mac01:dkv shashan2$ make all
go get -u github.com/golang/dep/cmd/dep
/Users/shashan2/distributed-kv-store/bin/dep ensure
go test -cover ./...
ok      dkv     0.040s  coverage: 0.0% of statements [no tests to run]
ok      dkv/api 2.045s  coverage: 50.9% of statements
go fmt ./...
go build -o /Users/shashan2/distributed-kv-store/target/dkv -v main.go
command-line-arguments
shashan2-mac01:dkv shashan2$ █
```