SDC Troubleshooting

Docker Diagram

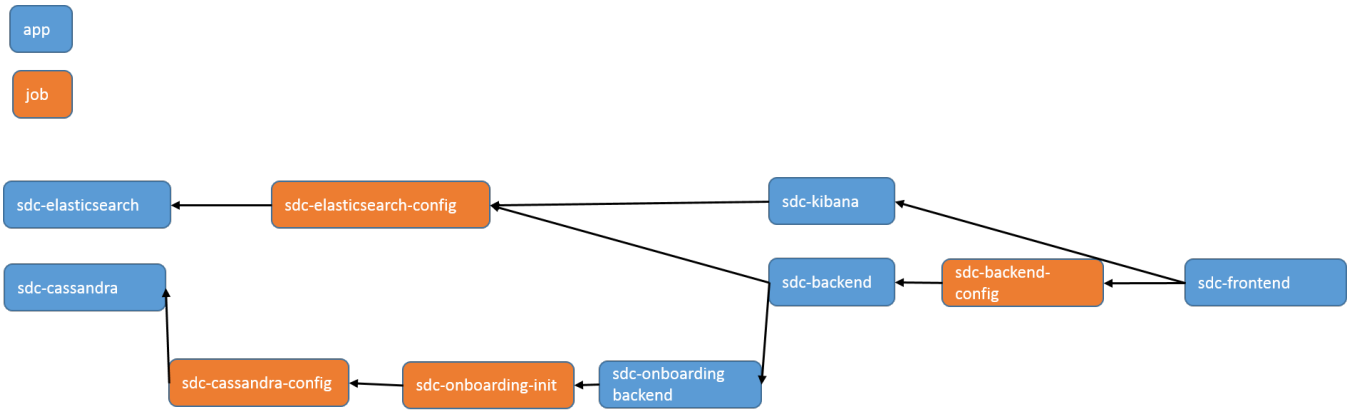
Amsterdam:

Docker name	Description
sdc-cassandra	The Docker contains our Cassandra server and the logic for creating the needed schemas for SDC. On docker startup, the schemes are created and Cassandra server is started.
sdc-elasticsearch	The Docker contains Elastic Search server and the logic for creating the needed mapping for SDC. On docker startup, the mapping is created and Elastic Search server is started.
sdc-kibana	The Docker contains the Kibana server and the logic needed for creating the SDC views there. On docker startup, the views are configured and the Kibana server is started.
sdc-backend	The Docker contains the SDC Backend Jetty server. On docker startup, the Jetty server is started with our application.
sdc-frontend	The Docker contains the SDC Fronted Jetty server. On docker startup, the Jetty server is started with our application.

Beijing:

Docker name	Description
sdc-cs	The Docker contains our Cassandra server . On docker startup the Cassandra server is started.
sdc-cs-init	The docker contains the logic for creating the needed schemas for SDC catalog server, On docker startup, the schemes are created.
sdc-cs-onboard-init	The docker contains the logic for creating the needed schemas for SDC onboarding server, On docker startup, the schemes are created.
sdc-es	The Docker contains Elastic Search server. On docker startup, Elastic Search server is started.
sdc-init-es	The Docker contains the logic for creating the needed mapping for SDC and the views for kibana. On docker startup, the mapping is created.
sdc-kibana	The Docker contains the Kibana server. On docker startup, the Kibana server is started.
sdc-onboard-BE	The Docker contains the onboarding Backend Jetty server. On docker startup, the Jetty server is started with the application.
sdc-BE	The Docker contains the catalog Backend Jetty server. On docker startup, the Jetty server is started with the application.
sdc-BE-init	The docker contains the logic for importing the SDC Tosca normative types and the logic for configuring external users for SDC external api's. on start, up the docker executes, the rest calls to the catalog server.
sdc-FE	The Docker contains the SDC Fronted Jetty server. On docker startup, the Jetty server is started with our application.

OOM/K8 deployment dependency map:



Connectivity Matrix

Docker name	API NAME	API purpose	protocol used	port number or range	TCP /UDP
sdc-cassandra		SDC backend uses the two protocols to access the cassandra	trift/async	9042/9160	TCP
sdc-elasticsearch		SDC backend uses the two protocols to access the ES	transport	9200/9300	TCP
sdc-kibana		the API is used to access the kibana UI	http	5601	TCP
sdc-onboard-backend		the APIs are used to access the onboarding functionalty	http/https	8081/8445	TCP
sdc-backend		the APIs are used to access the catalog functionalty	http/https	8080/8443	TCP
sdc-frontend		the APIs are used to access the SDC UI and to proxy requests to the SDC back end	http/https	8181/9443	TCP

Offered APIs

Container /VM name	API name	API purpose	protocol used	port number or range used	TCP /UDP
sdc-fe	/sdc1 /feproxy/*	Proxy for all the REST calls from the SDC UI	HTTP /HTTPS	8181/8443	TCP
sdc-be	/sdc2/*	Internal APIs used by the UI. The request is passed through the Front end proxy server	HTTP /HTTPS	8080/8443	TCP
sdc-be	/sdc/*	External APIs offered to the different components for retrieving information from the SDC Catalog. These APIs are protected by basic authentication.	HTTP /HTTPS	8080/8443	TCP
sdc~~onboarding~~ be	/onboarding-api/*	Internal APIs used by the UI.	HTTP /HTTPS	8081/8445	TCP

Status Information

Diagnostic:

We provide a health check script that can show the state of our application.
The script is located at /data/scripts/docker_health.sh.
The script is taken from our repository in LF on VM spin.
The script calls a REST API in the FE and BE server.

BE health Check URL:

```
http://<BE server IP>:<BE server port>/sdc2/rest/healthCheck
```

The Back end health check provides the following INFO, in case one of the components is down the server will fail requests:

type	section	description
general SDC info	"sdcVersion": "1.0.0-SNAPSHOT", "siteMode": "unknown",	This shows the current version of the Catalog application installed. The site mode is not used in the current version.

general Catalog info	{
	"healthCheckComponent": "BE",
	"healthCheckStatus": "UP",
	"version": "1.0.0-SNAPSHOT",
	"description": "OK"
	}
		This shows the current version of the catalog application installed.

Catalog sub components status		

Elastic Search	{
	"healthCheckComponent": "ES",
	"healthCheckStatus": "UP",
	"description": "OK"
	}
		This describes our connectivity to Elastic Search.

TITAN	{
	"healthCheckComponent": "TITAN",
	"healthCheckStatus": "UP",
	"description": "OK"
	}
		This describes our connectivity to and from the Titan client and the Cassandra server.

Cassandra	{ "healthCheckComponent": "CASSANDRA", "healthCheckStatus": "UP", "description": "OK" },	This describes the status of the connectivity from catalog to Cassandra

Dmaap	{
	"healthCheckComponent": "DE",
	"healthCheckStatus": "UP",
	"description": "OK"
	}
		This describes our connectivity to the Dmaap.

Onboarding	"healthCheckComponent": "ON_BOARDING", "healthCheckStatus": "UP", "version": "1.1.0-SNAPSHOT", "description": "OK",	This describes the state and version of the onboarding sub component
Onboarding sub component status		

Zusamen	{
	"healthCheckComponent": "ZU",
	"healthCheckStatus": "UP",
	"version": "0.2.0",
	"description": "OK"
	}
		this describes the version and status of the zusamen.

general Onboarding info	{
	"healthCheckComponent": "BE",
	"healthCheckStatus": "UP",
	"version": "1.1.0-SNAPSHOT",
	"description": "OK"
	}
		this describes the state and version of the onboarding sub component

Cassandra	{
	"healthCheckComponent": "CAS",
	"healthCheckStatus": "UP",
	"version": "2.1.17",
	"description": "OK"
	}
		This describes the connectivity status to Cassandra from the onboarding and the Cassandra version the onboarding is connected two.

The Front end server health check places a REST call to the Back end server to check the connectivity status of the servers.

the status received from the Backend server is aggregated in the Frontend health Check response.

in addition to the info retrieved from the BE the info of the Frontend server is added for the Catalog and Onboarding

FE health Check URL:

```
http://<FE server IP>:<FE server port>/sdc1/rest/healthCheck
```

type	section	description
general SDC info	in the main section	

Frontend	{
	"healthCheckComponent": "FE",
	"healthCheckStatus": "UP",
	"version": "1.1.0-SNAPSHOT",
	"description": "OK"
	}
		describe the version of the Catalog Frontend server

general Onboarding info	in the onboarding section	

	{
	"healthCheckComponent": "FE",
	"healthCheckStatus": "UP",
	"version": "1.1.0-SNAPSHOT",
	"description": "OK"
	}
		describe the version of the Onboarding Frontend server

Logging

server	location	type	description	rolling
BE	/data/logs/BE/2017_03_10.stderrout.log	Jetty server log	The log describes info regarding Jetty startup and execution	the log rolls daily
	/data/logs/BE/SDC/SDC-BE/audit.log	aplication audit	An audit record is created for each operation in SDC	rolls at 20 mb
	/data/logs/BE/SDC/SDC-BE/debug.log	aplication logging	We can enable higher logging on demand by editing the logback.xml inside the server docker. The file is located under: config/catalog-be/logback.xml. This log holds the debug and trace level output of the application.	rolls at 20 mb
	/data/logs/BE/SDC/SDC-BE/error.log	aplication logging	This log holds the info and error level output of the application.	rolls at 20 mb
	/data/logs/BE/SDC/SDC-BE/transaction.log	aplication logging	Not currently in use. will be used in future relases.	rolls at 20 mb
	/data/logs/BE/SDC/SDC-BE/all.log	aplication logging	On demand, we can enable log aggregation into one file for easier debugging. This is done by editing the logback.xml inside the server docker. The file is located under: config/catalog-be/logback.xml. To allow this logger, set the value for this property to true <property scope="context" name=" enable-all-log" value="false" /> This log holds all logging output of the application.	rolls at 20 mb
FE	/data/logs/FE/2017_03_10.stderrout.log	Jetty server log	The log describes info regarding the Jetty startup and execution	the log rolls daily
	/data/logs/FE/SDC/SDC-FE/debug.log	aplication logging	We can enable higher logging on demand by editing the logback.xml inside the server docker. The file is located under: config/catalog-fe/logback.xml. This log holds the debug and trace level output of the application.	rolls at 20 mb
	/data/logs/FE/SDC/SDC-FE/error.log	aplication logging	This log holds the Info and Error level output of the application.	rolls at 20 mb
	/data/logs/FE/SDC/SDC-FE/all.log	aplication logging	On demand we can enable log aggregation into one file for easier debuging, by editing the logback.xml inside the server docker. The file is located under: config/catalog-fe/logback.xml. To allow this logger set this property to true <property scope="context" name="enable-all-log" value="false" /> This log holds all the logging output of the application.	rolls at 20 mb

The logs are mapped from the docker to an outside path so that on docker failure the logs will still be available.

to change the log level in sdc:

1. access the docker for FE or BE example docker exec -it <docker id > bash
2. go to config/catalog-fe/logback.xml
3. open the file for editing.
4. in the file you can change the log level:

```
<root level="INFO">
        <appender-ref ref="ASYNC_ERROR" />
        <appender-ref ref="ASYNC_DEBUG" />
        <appender-ref ref="AUDIT_ROLLING" />
        <appender-ref ref="ASYNC_TRANSACTION" />
        <if condition='property("enable-all-log").equalsIgnoreCase("true")'>
                <then>
                        <appender-ref ref="ALL_ROLLING" />
                </then>
        </if>
</root>

<logger name="org.openecomp.sdc" level="INFO" />
```

5. change the root level to DEBUG to open all logs in SDC including the dependencies. (Note a lot of log output is created and it is hard to follow).
6. open the logger by a package to enable only sdc spcific logs.
7. it is important to note the opening the logs impacts the application performance so do not leave the system in debug level.
8. the log configuration is runtime editable so no restart is required for the docker, just save the file and that is enough.