MicroServices Onboarding in ONAP

- Introduction
- Onboarding
  - Data-Formats
  - Component-spec
- Design Phase
  - Template creation
  - Service Composition
- Runtime
- Code snippet to fetch configuration from Configbinding service
- DMAAP binding
- Sample configuration from DCAE Controller
  - Sample DR configuration structure

Introduction

This page provides a brief overview on the process involved in onboarding new MS onto DCAEGEN2 controller platform and integration with SDC, Policy and CLAMP for modelling, design and service instantiation. A more comprehensive documentation will be follow and delivered part of documentation project.

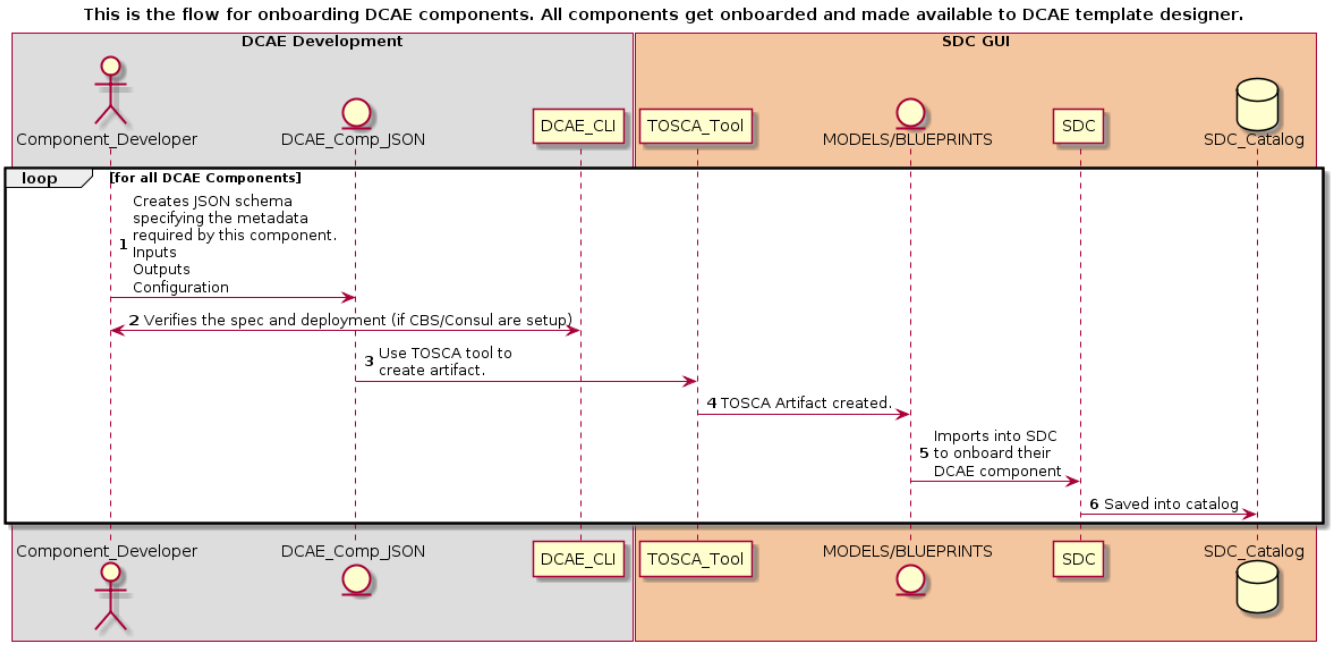
The DCAE Controller includes following components. The platform components will be deployed via platform level blueprint (not handled through onboarding steps noted here).

- Cloudify Manager
- Consul
- DeploymentHander
- Policy-Handler
- ServiceChangeHandler
- Inventory-API
- Postgres
- Registrator
- ConfigBinding Service
- CDAP Broker

For component/MS to be deployed into DCAE platform would go through below phases

- Onboarding Phase
- Design Phase
- Runtime

Onboarding



During this phase, each of the component owner should create a meta data describing the components. There are two main artifacts required to be defined

Data-Formats

Data formats are descriptions of data---they are the data contract between your component and other components. You need to describe the available outputs and assumed inputs of your components as data formats. These data descriptions are onboarded into SDC

Schema Definition

data-format-schema.json

Example: VES-4.27.2-data-format

VES-4.27.2-dataformat.json

Component-spec

The component specification is a JSON that is used to describe and configure your component. The component specification contains the following top-level groups of information:

- Component metadata
- Component interfaces including the associated data formats
- Configuration parameters
- Auxiliary details
- List of artifacts

Schema Definition

component-spec-schema.json

Example: VESCollector

vescollector-componentspec.json

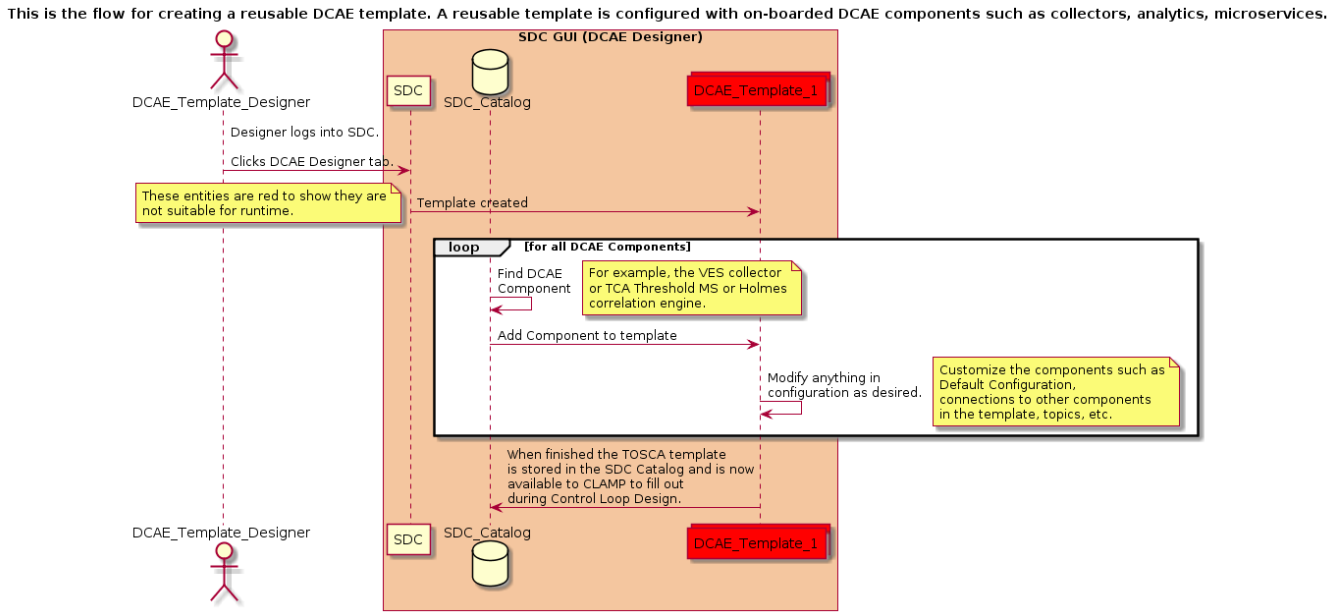
The developers will upload the spec (data-formats, component spec) developed into catalog using dcae_cli/tosca tool. Once validated, spec's will be published into SDC catalog so they can be used for service creation.

Design Phase

Design comprises of two steps.

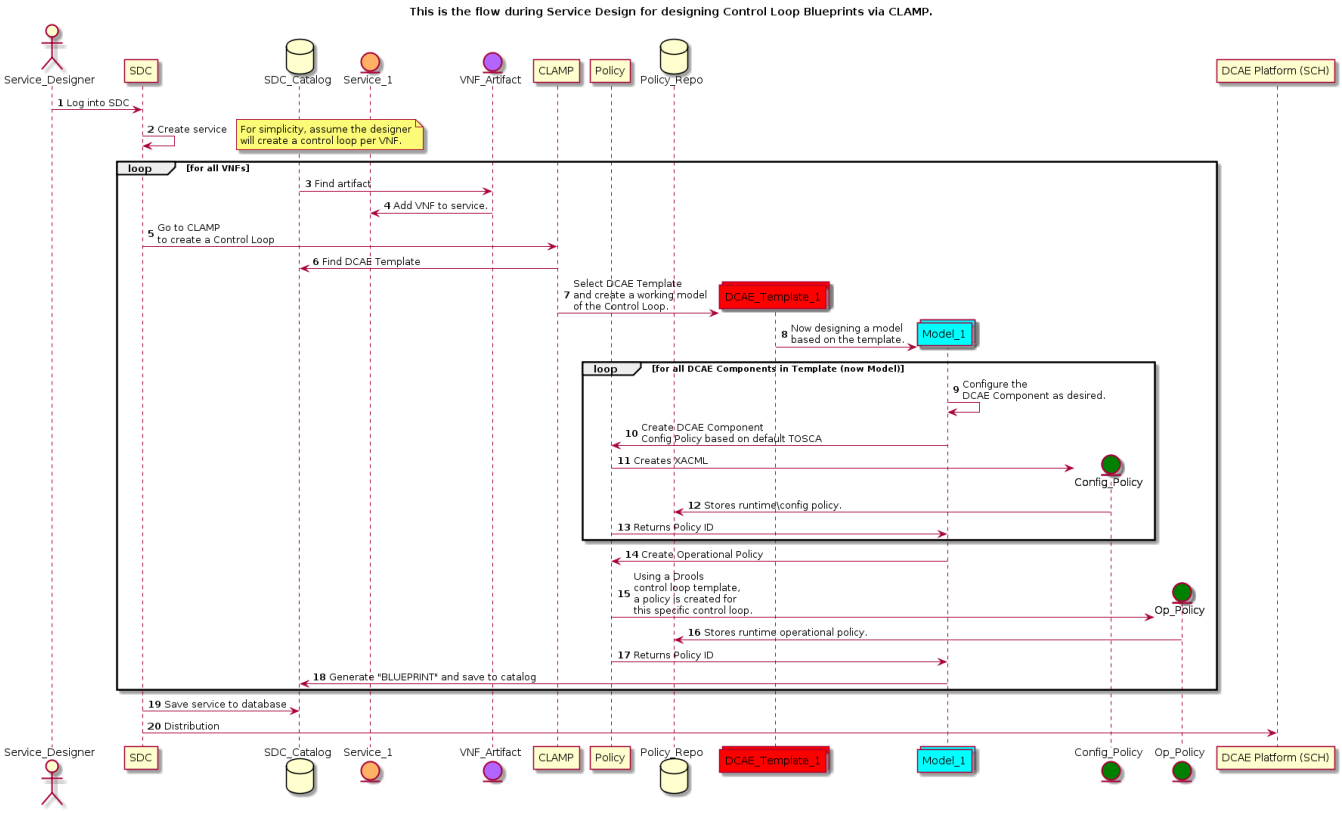
Template creation

The TOSCA models generated from onboarding is used here to do this composition. Using SDC designer tool – individual reusable templates are created and stored in catalog.



Service Composition

Once the designer template is created, they can be used for Service composition (in SDC/CLAMP) to generate a Cloudify blueprint.



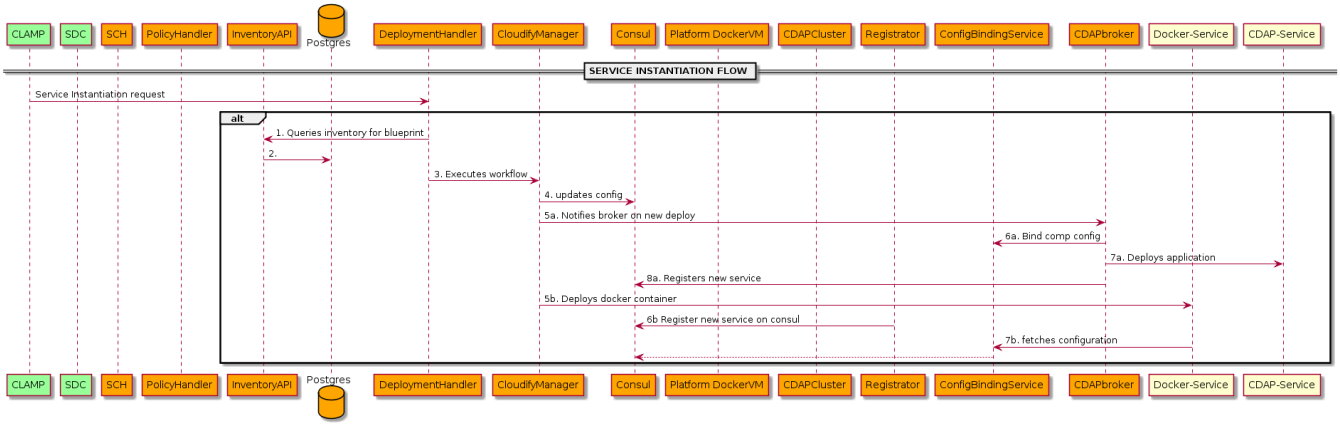
Once approved, SDC sends notification via an event onto a message router topic for DCAE-ServiceChangeHandler, which queries SDC to retrieve the blueprint and store into DCAE inventory.

Runtime

Instantiation flow is triggered by update from ASDC/CLAMP where the DCAE controller deploys the components associated with service which is identified in the cloudify blueprint.

Once Dispatcher receives the triggering event (e.g. instantiation of service/component), it gets the corresponding blueprint from Inventory and then loads into Cloudify manager where the blue print is executed to deploy the service components.

The complete flow with DCAE platform is depicted below.



Code snippet to fetch configuration from Configbinding service

For Docker containers, the configuration identified part of MS component-spec is retrieved from Configbinding service during application start-up and when notified by controller on a configuration change. The docker container part of init script - should query configbindingservice and fetch the configuration.

Below is code snippet which does the configuration fetch in two steps

- Identifies ConfigbindingService ip and port by querying Consul
- Queries ConfigbindingService and retrieves the configuration into a file.

```
if (env.containsKey("CONSUL_HOST") && env.containsKey("CONFIG_BINDING_SERVICE")
    && env.containsKey("HOSTNAME")) {
    log.info(">>>Dynamic configuration to be fetched from ConfigBindingService");
    url = env.get("CONSUL_HOST") + ":8500/v1/catalog/service/" + env
        .get("CONFIG_BINDING_SERVICE");

    retString = executecurl(url);
    // consul return as array
    JSONTokener temp = new JSONTokener(retString);
    JSONObject cbsjobj = (JSONObject) new JSONArray(temp).get(0);

    String urlPart1 = null;
    if (cbsjobj.has("ServiceAddress") && cbsjobj.has("ServicePort")) {
        urlPart1 =
            cbsjobj.getString("ServiceAddress") + ":" + cbsjobj.getInt("ServicePort");
    }

    log.info("CONFIG_BINDING_SERVICE DNS RESOLVED:" + urlPart1);
    url = urlPart1 + "/service_component/" + env.get("HOSTNAME");
    retString = executecurl(url);

    JSONObject jsonObject = new JSONObject(new JSONTokener(retString));
    try (FileWriter file = new FileWriter(configFile)) {
        file.write(jsonObject.toString());

        log.info(
            "Successfully Copied JSON Object to file /opt/app/KV-Configuration.json");
    } catch (IOException e) {
        log.error(
            "Error in writing configuration into file /opt/app/KV-Configuration.json "
                + jsonObject, e);
    }
}
```

The data is represented as key-value pair gets written into a file within container above. The MS should parse the configuration and apply within MS as required.

For new JVM code you are advised to use cbs-client from DCAE SDK instead of manually implementing the code above. For python clients - onap_dcae_cbs_docker_client module is available under Global PyPI.

DMAAP binding

The dmaap topic stream are identified in compenent_spec as "subscribe" and "publish" streams. Each topic should be identified by the component developer using unique config_key. When the configuration gets pushed into the container by controller, the topic configuration is pushed under below structure.

"streams_publishes": {

"config_key1": {

"type": "message_router",
"aaf_password": "value",
"dmaap_info": {
"location": "loc",
"client_id": "id",
"client_role": "aafrole",
"topic_url": "topic-url"
},
"aaf_username": "user"

} ,

"config_key2": { similar structure }

},
"streams_subscribes": {

"config_key3": { similar structure} ,

"config_key4": [http stuff]

}

The config_key defined will map the value specified in the component_spec. For an anonymous (non-secure) topic, the username/password field will be defaulted to null.

Sample configuration from DCAE Controller

Below is sample Configuration for VESCollector retrieved from DCAE platform (configbinding service).

```
{
    "collector.schema.file": "./etc/CommonEventFormat_27.2.json",
    "collector.service.port": 8080,
    "collector.dmaap.streamid": "fault=sec_fault,roadm-sec-to-
hp|syslog=sec_syslog|heartbeat=sec_heartbeat|measurementsForVfScaling=sec_measurement|mobileFlow=sec_mobileflow|ot
her=sec_other|stateChange=sec_statechange|thresholdCrossingAlert=sec_thresholdCrossingAlert",
    "collector.schema.checkflag": 1,
    "tomcat.maxthreads": "200",
    "collector.keystore.passwordfile": "/opt/app/dcae-certificate/.password",
    "streams_subscribes": {},
    "services_calls": {},
    "collector.inputQueue.maxPending": 8096,
    "header.authflag": 0,
    "collector.keystore.file.location": "/opt/app/dcae-certificate/keystore.jks",
    "collector.service.secure.port": -1,
    "header.authlist": "userid1,base64encodepwd1|userid2,base64encodepwd2",
    "collector.keystore.alias": "dynamically generated",
    "streams_publishes": {
            "sec_measurement": {
                "type": "message_router",
                "aaf_password": "aaf_password",
                "dmaap_info": {
                    "location": "mtl5",
                    "client_id": "111111",
                    "client_role": "com.att.dcae.member",
                    "topic_url": "https://mrlocal:3905/events/com.att.dcae.dmaap.FTL2.SEC-MEASUREMENT-OUTPUT"
                },
                "aaf_username": "aaf_username"
            },
            "sec_fault_unsecure": {
                "type": "message_router",
                "aaf_password": null,
                "dmaap_info": {
                    "location": "mtl5",
                    "client_id": null,
                    "client_role": null,
                    "topic_url": "http://ueb.global:3904/events/DCAE-SE-COLLECTOR-EVENTS-DEV"
                },
                "aaf_username": null
            },
            "sec_measurement_unsecure": {
                "type": "message_router",
                "aaf_password": null,
                "dmaap_info": {
                    "location": "mtl5",
                    "client_id": null,
                    "client_role": null,
                    "topic_url": "http://ueb.global:3904/events/DCAE-SE-COLLECTOR-EVENTS-DEV"
                },
                "aaf_username": null
            },
            "sec_fault": {
                "type": "message_router",
                "aaf_password": "aaf_password",
                "dmaap_info": {
                    "location": "mtl5",
                    "client_id": "222222",
                    "client_role": "com.att.dcae.member",
                    "topic_url": "https://mrlocal:3905/events/com.att.dcae.dmaap.FTL2.SEC-FAULT-OUTPUT"
                },
                "aaf_username": "aaf_username"
            }
    }
}
```


Sample DR configuration structure


```
{
```

```
"streams_subscribes": {
 "DCAE_GUEST_OS": {
  "type": "data_router",
  "dmaap_info": {
   "username": "xyz",
   "password": "abc",
   "location": "mtn23",
   "delivery_url": "https://dr.global:8666/DCAE_SAM_GUEST_OS",
   "subscriber_id": "811"
  }
 },
 "DCAE_RAW_DATA": {
  "type": "data_router",
  "dmaap_info": {
   "username": "abc",
   "password": "xyz",
   "location": "mtn23",
   "delivery_url": "https://dr.global:8666/DCAE_CEILOMETER_RAW_DATA",
   "subscriber_id": "812"
  }
 },
 "sec-measurement-output": {
  "type": "message_router",
  "aaf_password": "aaf_password",
  "dmaap_info": {
   "topic_url": "https://mr.hostname:3905/events/com.att.dcae.dmaap.SEC-MEASUREMENT-OUTPUT-v1",
   "client_role": "com.att.dcae.member",
   "location": "mtn23",
   "client_id": "1111"
  },
  "aaf_username": "aaf_username"

 }

},

"streams_publishes": {

 "DCAE_VOIP_PM_DATA": {
  "type": "data_router",
  "dmaap_info": {
   "username": "abc",
   "log_url": "https://dcae-drps/feedlog/206",
   "publish_url": "https://dcae-drps/publish/206",
   "location": "mtn23",
   "password": "xyz",
   "publisher_id": "206.518hu"

  }
 },

 "DCAE_GUEST_OS_O": {
  "type": "data_router",
  "dmaap_info": {
   "username": "axyz",
   "log_url": "https://dcae-drps/feedlog/203",
   "publish_url": "https://dcae-drps/publish/203",
   "location": "mtn23",
   "password": "abc",

   "publisher_id": "203.2od8s"
  }
 },
```

```
  "DCAE_PM_DATA": {
   "type": "data_router",
   "dmaap_info": {
    "username": "xyz",
    "log_url": "https://dcae-drps/feedlog/493",
    "publish_url": "https://dcae-drps/publish/493",
    "location": "mtn23bdce2",
    "password": "abc",
    "publisher_id": "493.eacqs"
   }
  }
 }

}
```