DCAE Multi-Site Support in ONAP Dublin Release

2019-04-24/Jack Lucas

- Goal
  - Run data collection and analysis close to VNFs/PNFs being monitored
  - Use the current Cloudify-based deployment mechanism
  - Continue to run a single DCAE controlled from a central site
    - 1 instance of Cloudify Manager, deployment handler, Consul, config binding service, etc.
- Assumptions
  - Central site with a full ONAP instance
  - Remote sites are independent k8s clusters
    - Isolated private cluster network
    - Independent cluster DNS
  - Network connectivity between k8s clusters at the k8s host level
  - No “central” components deployed into remote sites
- Elements of the solution
  - Provide a way to specify target deployment site in a blueprint
  - Make k8s cluster information for sites available to the k8s Cloudify plugin
  - Allow components in remote sites to access services running in the central site
    - DCAE collection and analytics components typically need Consul, config binding service, logstash

Specifying Target Deployment in a Blueprint

- Add a `location_id` property to k8s node types
  - Value is a string that identifies a k8s cluster
  - Optional – default is to deploy into central site
- Blueprints can be site-independent by specifying `location_id` as an input

```yaml
tosca_definitions_version: cloudify_dsl_1_3

description: >
  Simple blueprint to launch nginx as a "service component"

imports:
  - http://www.getcloudify.org/spec/cloudify/3.4/types.yaml
  - https://nexus.onap.org/service/local/repositories/raw/content/org.onap.dcaegen2.platform.plugins/R4/k8splugin/1.4.12/k8splugin_types.yaml

inputs:
  location:
    type: string
    default: site-00

node_templates:
  web_server:
    type: dcae.nodes.ContainerizedServiceComponent
    properties:
      service_component_type: 'nginx-web'
      location_id: {get_input: location}
      image: nginx
      docker_config:
        healthcheck:
          type: "http"
          endpoint: "/"
    interfaces:
      cloudify.interfaces.lifecycle:
        start:
          inputs:
            ports:
              - '80:0'
```

Make k8s Cluster Information Available to Plugin

- Preferred Solution
  - When new k8s cluster is brought up, record information about it in A&AI
  - Map site name to information about the cluster (address, credentials)
  - Entities that need to deploy/manage components in remote sites query A&AI for needed information
  - Doesn’t exist yet
- Interim Solution
  - Use the k8s “kubeconfig” file as the storage for information about sites
    - One k8s “context” for each k8s cluster
    - Site name is used as the context name
  - Store “kubeconfig” in a k8s ConfigMap in the central site cluster
    - OOM Helm deployment creates an empty ConfigMap
    - Cloudify Manager init container adds information for the “central” location
    - Additional sites added by manually editing the ConfigMap
    - Cloudify Manager mounts ConfigMap so plugin can access the information
    - Other applications could mount the ConfigMap as well

Allow Components to Access Central Site Services

- DCAE components use ONAP and DCAE services running in central site
  - Consul, config binding service, logstash – at least
  - These services are exposed outside the central k8s cluster as NodePort Services.
- Two possible approaches to providing access
  - Make components in remote sites aware of central cluster addresses and NodePort mappings
  - Make central services appear to be local in each remote site
  - This implementation takes the second approach
- Making the central services appear to be local
  - Run an instance of nginx in each remote cluster as a proxy
    - Routes traffic arriving local on the internal ports for Consul (8500), config binding service (10000), and logstash (5044) to the external ports at the central site
  - Create k8s ClusterIP Services for Consul, config binding service and logstash in each remote cluster
    - Use the internal port addresses for the services
    - Point the Services to the local nginx proxy
  - Remote site proxy and services are deployed with a Helm chart
    - `values.yaml` specifies:
      - IP addresses of hosts in central local
      - Service names and port mappings (can be extended beyond the three mentioned here)
  - What about DMaaP?
    - Could use proxy to expose DR and MR running in central site
    - DMaaP has a way to deploy into remote sites
    - Coordinating DMaaP and DCAE is TBD

Central and Remote Sites