Controller Flow: Create Cluster

Overview:

This document describes a high level flow for a create cluster invocation in the Savanna controller. It is assumed that a mechanism will exist to plug cloud and Hadoop providers into the Savanna controller. In addition, it is expected that there will be a controller element responsible for invoking these plugins. Described is the interaction of the controller with the cloud and Hadoop providers with the intent of identifying the interaction points and possible interfaces. The focus here is primarily on the Hadoop provider functionality and the services that it requires. The create cluster invocation results in a number of VM's being provisioned by the cloud provider which are then used by the Hadoop provider to install and start a Hadoop cluster.

- Controller receives create cluster request
  - received arguments
    - cloud_spec
      - specifies cloud provider (e.g. openstack …)
      - provider specific details used to provision vm's
      - vm_image(s), flavor(s), node_count, etc.
    - cluster_spec
      - specifies Hadoop provider
      - provider specific cluster details
      - cluster, services, hosts, host_components, etc.
- Controller calls provision_nodes(cloud_spec) on cloud provider
  - cloud provider provisions vm's based on cloud_spec
  - vm's are provisioned concurrently
  - blocks until network info (ip addrs) are available for all nodes
  - returns cloud_ctx
    - general cluster information
    - servers[]
      - server specific information such as hostname, public/private ip addrs, vm_image, flavor, etc.
      - functionality related to interacting with provisioned VM's
        - install_package(package)
          - yum -y, zypper --non-interactive, …
        - execute(command)
          - execute a command on a server
        - execute_interactive(command, prompts)
          - execute a command that requires interactive responses
        - open_file(file, mode)
          - returns a file pointer to a remote file
- Controller calls create_cluster(cloud_ctx, cluster_spec) on Hadoop provider
  - Provider matches "host groups" to servers in cloud_ctx
    - examples would be "master and "slave"
    - match based on provided "qualifiers" for each group in cluster_spec (e.g. match "master" memory qualifier with a VM having that amount of memory or more)
  - HDP specific details

- install/confirm various epel packages on nodes
  - cloud_ctx.server[i].install('epel-release')
  - …
- install/setup/start Ambari components
  - Server only installed on master, determine "master" host (assuming master host is resolved to index '0')
  - install ambari-server package
    - cloud_ctx.server[0].install('ambari-server')
  - setup ambari server
    - cloud_ctx.server[0].execute_interactive('ambari-server setup', prompts)
  - start ambari-server
    - cloud_ctx.server[0].execute('ambari-agent start')
  - install/start agent on all hosts in same manner
- update various configurations on hosts
  - f = cloud_ctx.servers[i].open_file('/my_config.cfg', 'r+')
  - f.write('some configuration')
  - …
- interact with Ambari via REST api
  - invoked via savanna_ctx
    - context passed to provider during init
    - provides services to providers(persistence, …)
    - savanna_ctx.invoke_rest(request)
  - install Hadoop services
  - start Hadoop services
- persist cluster-> master_host mapping
  - persisted using savanna_ctx
  - savanna_ctx.persist(cluster_name, master_server)
- …