Plugin API

Functions

get_versions()

Returns all versions of Hadoop that could be used with the plugin. It is responsibility of the plugin to make sure that all required images for each hadoop version are available, as well as configs and whatever else that plugin needs to create the Hadoop cluster.

Returns: list of strings - Hadoop versions

Example Return Value: (“Apache Hadoop 1.1.1”, “CDH 3”, “HDP 1.2”)

get_configs(hadoop_version)

Lists all configs supported by plugin with descriptions, defaults and targets for which this config is applicable.

Returns: list of **config**s

Example Return Value: ((“heap size”, “512”, true, “jt”))

get_supported_node_process(hadoop_version)

Lists of all supported NodeProcesses for a given Hadoop version.

Returns: list of strings - node processes

Example Return Value: (“jt”, “nn”, “tt”, “dt”)

convert(cluster, file)

Provides plugin with ability to create cluster based on plugin-specific config. Savanna expects plugin to fill in all the required fields. See “Cluster Lifecycle for Config File Mode” section below for clarification.

validate_cluster(cluster)

Validates **user_input**s in a given **cluster**. Returns empty list if all inputs are correct. Otherwise for each incorrect input function should return **validation_error** with meaningful content.

For each **user_input** the function must check that it is applied to the correct Node Type, i.e. that corresponding **config.applicable_targets** and Node Type has one of the node

processes in common. Also function should check that the provided value is correct for the given **config**.

Returns: list of **validation_error**s

*Example Return Value***:** ((“heap size”), (“Heap size is a required field and must be specified”), (“mapred.task.timeout”, “The parameter must be int”))

get_infra(cluster)

Plugin has chance to change cluster description here. Specifically, plugin

- must specify image for VMs
- could change VMs specs in any way it needs. For instance, plugin can ask for additional VMs for the management tool.

*Returns***:** None

configure_cluster(cluster)

Configures cluster on provisioned by savanna VMs. In this function plugin should perform all actions like adjusting OS, installing required packages (including Hadoop, if needed), configuring Hadoop, etc.

Returns: None

start_cluster(cluster)

Start already configured cluster. This method is guaranteed to be called only on cluster which was already prepared with **configure_cluster(...)** call.

Returns: None

scale_cluster(cluster, remaining_vm_specs, new_vm_specs, delete_vm_specs)

To be changed

Scales cluster - adds/removes nodes to/from active cluster. This function should configure new nodes and attach them to cluster. Additionally here plugin can make some cleanup on VMs that will be deleted.

Returns: None

on_terminate_cluster(cluster)

When user terminates cluster, Savanna simply shuts down all the cluster VMs. This method is guaranteed to be invoked before that, allowing plugin to do some clean-up.

Returns: None

Objects

All fields are strings unless specified otherwise. **cluster** and **node_group** have 'extra' field allowing plugin to persist any complementary info about the cluster.

config

Describes a single config parameter.

name
description
type
    Type could be string, integer, enum, array of [int, string]
default_value
is_optional
applicable_targets: list of strings
    The target could be either a node_process, 'node' or 'cluster'

user_input

Value provided by user for a specific **config**.

config_name
value

instance

An instance created for the cluster

id
ip
credentials
extra

node_group

Specifies group of nodes within a cluster.

name
    Helps uniquely identify the group
flavor
    OpenStack flavor which is used to launch instances of this group

image
: Image from Glance which is used to launch instances of this group

node_processes: list of strings
: List of Hadoop processes which run on nodes of this group

node_configs: list of **user_input**s
: List of configs provided by user. The configs either are for Hadoop processes running on this group, or they relate to the plugin-specific node configuration.

anti_affinity_group
: That parameter will be used to control node placement for DN nodes. TBD

count: int
: Number of instances in that group

instances: list of **instance**s
: List of instances create for that group

extra

cluster

Contains all relevant info about cluster. This object is is provided to the plugin for both cluster creation and scaling. The “Cluster Lifecycle” section below further specifies which fields are filled at which moment

name

plugin_name

default_image
: Image specified by user. Plugin could use it as a base image for cluster nodes

cluster_configs: list of **user_input**s
: List of cluster-wide configs

node_groups: list of **node_group**s

extra

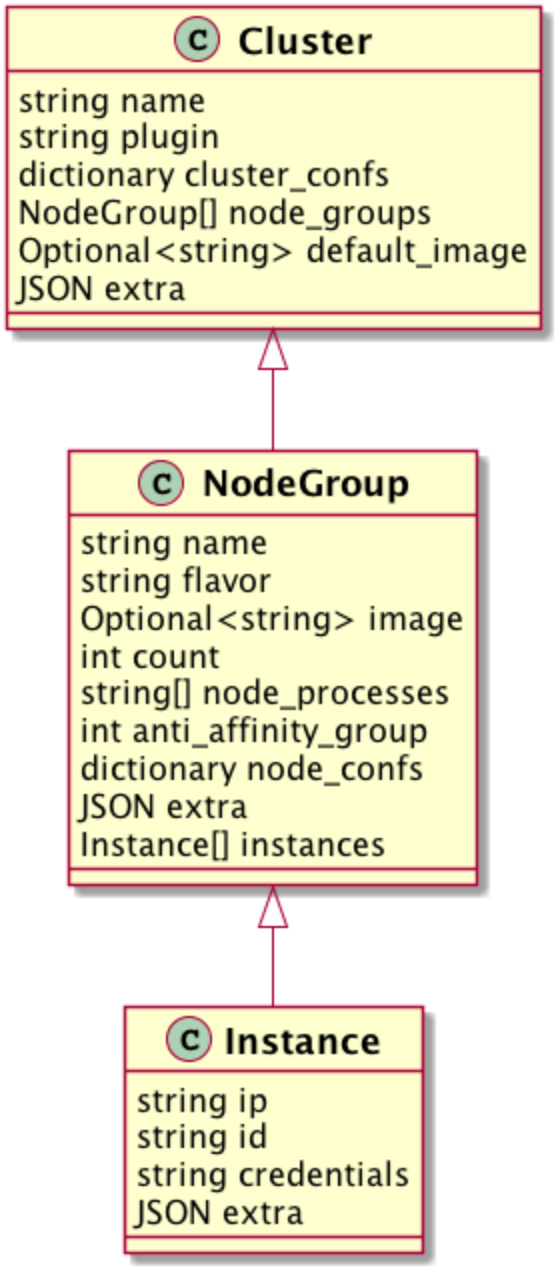
validation_error

Describes what is wrong with one of the values provided by user.

config_name

error_message

Cluster object schema:



Cluster Lifecycle for Templates Mode

The ‘cluster’ object is passed as an argument to several functions. It could be used to both get cluster configuration and change it. During these invocations the object has some fields blank.

validate_cluster(...) and get_infra(...) both receive cluster with the same set of fields set. The validate_cluster(...) must not change the cluster object, while get_infra(...) can (and for some fields must) do some adjustments. Below is the list of fields which will not be set at the time of invocation. All comments stating that plugin can or must adjust some field are for get_infra(...) function only:

- image - plugin must set that field. It is assumed that plugin set this field to either default_image from parent **cluster** object or an image obtained from Image Registry.
- instances. At that point cluster instances are not yet created.

The plugin could completely rewrite node_groups field. It is assumed that the plugin will use this ability to add specs for management nodes if needed or something like that.

configure_cluster(...) and start_cluster(...) both receive ‘cluster’ object with all fields set

Cluster Lifecycle for Config File Mode

The previous section covered cluster creation with templates. This section elaborates on creating cluster from provider-specific config file.

In this mode, user starts creating cluster with providing provider-specific configuration file. Savanna passes that file to the plugin’s convert(...) method along with empty ‘cluster’ object. Plugin is expected to parse cluster configuration from the file and populate ‘cluster’ object with cluster topology.

After convert(...) returns, Savanna shows the cluster dialog to the user. In the dialog user can change number of nodes and flavors of node groups. See cluster mockups on the wiki for clarity. After user submits the form, the flow goes through the same cycle as for Templates mode, except one thing:

Neither cluster_configs nor node_configs will be provided to the plugin. It is up to the plugin if it uses these fields. Alternatively plugin can save the config file in extra field and use it later.

Cluster Lifecycle Colourgram

Below is the description of Savanna core and plugin’s responsibility for cluster object for each phase of its lifecycle.
Color legend for fields:

colour	**is set**	**modifiable**	**must be specified?**
black	yes	-	-
grey	-	-	-
underlined blue	yes	yes	-
blue	-	yes	-
red	-	yes	yes

convert(cluster)

cluster:
- name
- plugin_name
- default_image
- cluster_configs
- node_groups:
  - name
  - flavor
  - image
  - node_processes
  - node_configs
  - anti_affinity_group
  - count
  - instances:
    - id
    - ip
    - credentials
    - extra
  - extra
- extra

validate_cluster(cluster)

cluster:
- name
- plugin_name
- default_image
- cluster_configs
- node_groups:
  - name
  - flavor
  - image
  - node_processes
  - node_configs
  - anti_affinity_group
  - count
  - instances:
    - id
    - ip
    - credentials
    - extra
  - extra

extra

get_infra(cluster)

cluster:

name
plugin_name
default_image
cluster_configs
node_groups:
name
flavor
image
node_processes
node_configs
anti_affinity_group
count
instances:
id
ip
credentials
extra
extra
extra

configure_cluster(cluster), start_cluster(cluster)

cluster:

name
plugin_name
default_image
cluster_configs
node_groups:
name
flavor
image
node_processes
node_configs
anti_affinity_group
count
instances:
id
ip
credentials
extra
extra

extra

Image Registry API

A component helping plugin find suitable images by some criteria. All search is based on tags. A tag is just a string. Each image could have several tags simultaneously.

set_description(image, os_description, hadoop_version, extra)

Sets human-readable information for image, for example, “Ubuntu 13.04 x86_64, Apache Hadoop 1.1.1, Java 1.7u21”
Returns: None

tag_image(image, tags)

Adds tags to image
Returns: None

untag_image(image, tags)

Removes tags from image
Returns: None

get_image_tags(image)

Queries all tags for the given image
Returns: list of strings - image tags

get_image_by_tags(image, tags)

Queries images having all of the specified tags
Returns: list of strings - images’ ids

VM Manager

A pack of low-level helpers to help plugin interact with vms

execute(command)

Executes command in shell on VM via ssh (non-interactive)

copy_to_vm(filename)

Copies file to VM via ssh

copy_from_vm(filename)

Copies file from vm via ssh

Additionally we are thinking about adding some helpers for some high-level actions, for example:

install package (using apt-get/yum ?)