Arenadata™ Cluster Manager

Версия - adcm-pipeline

Software Development Kit

2018 Arenadata™

Оглавление

1 Bundle **2**
- 1.1 Cluster Bundle . . . 2
- 1.2 Host Bundle . . . 2

2 Config File **3**
- 2.1 Prototype . . . 3
- 2.2 Actions . . . 4
- 2.3 Components . . . 4
- 2.4 User Config . . . 5
- 2.5 Export . . . 8
- 2.6 Import . . . 8
- 2.7 Upgrade . . . 9

3 Ansible module: adcm_config **10**

Глава 1

Bundle

Bundle is a minimum unit of delivery for ADCM. Bundle should include at least one *Config File*. Also bundle could include more config files, ansible script files and any other required files.

All directories and files realtive to one bundle should be compressed by **tar** command to one bundle file.

This bundle file can be uploaded to ADCM via

POST /api/v1/stack/upload/

and loaded into ADCM database via

POST /api/v1/stack/load/

When loaded, ADCM umcompress bundle file, search for every file with named **config.yaml** and treat them as *Config File*.

Bundle could be one of two types:

1.1 Cluster Bundle

Cluster bundle should include exactly once definition of cluster prototype and zero, one or more definitions of service prototypes. These definitions can be in one or many config files.

1.2 Host Bundle

Host bundle should include exactly once definition of host prototype.

Глава 2

Config File

Config file is a way to decribe how ADCM should run ansible playbooks and which parameters should be asked from and user.

Format of config file is YAML. For additional information about YAML syntax it's better to start digging from https://en.wikipedia.org/wiki/YAML.

That file have to be named as config.yaml or config.yml. Multiple config.yaml files allowed in a bundle.

Every config is a list of objects called prototypes.

2.1 Prototype

Prototype is a description of one object.

Every prototype should be one of the following types:

- cluster
- service
- host

Every prototype has following mandatory properties:

- type
- name
- version

So minimal working prototype should be like that:

```
---
- type: cluster
  name: control
  version: 1
```

There is optional description field

```
---
- type: cluster
  name: control
```

```
  version: 1
  description: "Monitoring and Control Software"
```

2.2 Actions

Every prototype could have an actions. Actions should be described as follows:

```
---
- type: cluster
  name: control
  version: 1
  description: "Monitoring and Control Software"

  actions:
    install:
      type: job
      script_type: ansible
      script: ansible/site.yaml
      params:
        ansible_tags: install
      states:
        available:
          - created
        on_success: installed
        on_fail: created
      config:
         quorum:
            type: integer
```

You could see one action called "**install**". That action has job type and that is ansible playbook that resides in directory <bundle_root>/ansible/site.yaml

You could also pass any additional parameter to ansible call. They will be used in ansible-playbook calls.

One special case is parameter "ansible_tags". That parameter turn to be –tags parameter of ansible-playbook call

Action can have a **config** optional statment. It can define a list of values as in *User Config*. When action is run via UI, all this values will be asked from user before action start.

In action ansible script you can later refer this values:

```
{{ job.config.quorum }}
```

2.3 Components

Service prototype could have a components. Component is a mean of placing service components on hosts in cluster.

```
---
- type: service
  name: server
  version: 1
```

```
  components:
    master:
      constraint: [1,2]
    node:
      constraint: [+]
    secondary:
      constraint: [0,1]
```

Every component can has an optional constraint field. It describe how many instances of this component should be installed in one cluster.

- [1] - exactly once component shoud be installed
- [0,1] - one or zero component shoud be installed
- [1,2] - one or two component shoud be installed
- [0,+] - zero or any more component shoud be installed (**default** value)
- [1,+] - one or any more component shoud be installed
- [+] - component shoud be installed on all hosts of cluster

2.4 User Config

There are a number of user editable values that could be represented in UI and related to every object.

```
---
- type: cluster
  name: control
  version: 1
  description: "Monitoring and Control Software"

  config:
    repos:
      ads:
        type: string
        default: https://ci.arenadata.io/artifactory/list/ads-centos7-x64-ads/
        required: no
```

In the example you could see parameter “ads” in group “repos”. That parameter is not required. It means that it is posible for user to leave it empty.

Group is not mandatory. So you could place your parameter on highest level:

```
---
- type: cluster
  name: control
  version: 1
  description: "Monitoring and Control Software"

  config:
    ads:
      type: string
      default: https://ci.arenadata.io/artifactory/list/ads-centos7-x64-ads/
      required: no
```

Group is just a visual representation in UI.

2.4.1 Common config parameters

Name	Mandatory	Values	Default
type	yes	string/text/boolean/option/integer/float/password/file	
default	no	user defined	per type
required	no	yes/no	yes
read_only	no	see below	
writable	no	see below	

If required set to "yes" it is required to have non empty and valid value for a variable

2.4.2 Types of config parameters

String

Arbitrary string. The size is not limited. If default not explicitly set, default value would be empty string ""

Text

The same as string, but will be displayed as multi line text in user interface

Boolean

Boolean value. True of False

Integer

Integer number. If default not explicitly set, default value would be 0. You can use optional **min** and **max** parameters to limit allowed values

```
config:
  memory_size:
    type: integer
      default: 16
      min: 2
      max: 64
```

Float

Float number. If default not explicitly set, default value would be 0.0. You can use optional **min** and **max** parameters to limit allowed values

```
config:
  average_load:
    type: float
      default: 1.0
      min: 0.5
      max: 3.5
```

Password

The same as string type, but it dipsplays as password field in user interface

Option

Key-value dictionary type. Keys are showed in user interface, corresponding values are stored in DB

```
config:
  protocol:
    type: option
    option: {http: 80, https: 443}
    default: 80
```

File

The content of this field is stored as file on file system. When you refernce to this variable in ansible script it return full path to this file.

If default value is set, it should be name of file relative to bundle root. When object is created, content of this file will be read to to the value of specified config parameter and stored on file system. (Distinct file for each object).

User can edit content of this file via user interface. File will be updated after each config saving.

Can be usable to story ssh private keys for ansible scripts

```
- type: host
  name: ssh
  version: 1

  config:
    ansible_private_key_file:
      type: file
```

2.4.3 Read Only / Writable

You can make any config parameter read only (protectected from change by user or API) for specified states of cluster/host/service

Examples:

Parameter quorum will be read only (protectected from change) when service state is “installed” or “runnig”:

```
config:
  quorum:
    type: integer
    read_only: [installed, runnig]
```

Parameter quorum will be read only (protectected from change) for all service states except “created”:

```
config:
  quorum:
    type: integer
    writable: [created]
```

Parameter quorum will be read only (protectected from change) for all service states

```
config:
  quorum:
    type: integer
    default: 4
    read_only: any
```

2.5 Export

If you want to share some parts ot config with other clusters, you can use **export** statment. In **export** you can specify any number of first level config entries. You can export config in cluster or/and in service

```
---
- type: service
  name: Hadoop
  version: 2.1

  config:
     core-site:
        param1:
           type: string
        param2:
           type: int
     quorum:
        type: int
        default: 3

  export:
      - core-site
      - quorum
```

2.6 Import

To use exported parts of config from other cluster you should use **import** statment in your cluster. In **import** you should specify name and acceptable versions of exported cluster (or service in cluster)

```
---
-
  type: cluster
  name: Hive
  version: 4

  import:
     Hadoop:
        versions:
           min: 1.8
           max: 2.5
```

When you bind you cluster and another cluster/service via ADCM user inteface, you can refer exported config parameters in your ansible scripts like this:

```
{{  cluster.import.Hadoop.core-site.param2  }}
{{  cluster.import.Hadoop.quorum  }}
```

2.7 Upgrade

Cluster or host prototype could have an **upgrade** statment. This statment describe how to upgrade from one version of cluster to another. Prototype could have a few upgrade items to discribe upgrade from different versions. Each item should include:

- versions - interval of versions, appropriate for upgrade
- name (optional)
- description (optional)
- states:
- available - list states of cluster, permited to upgrade
- on_success - state of cluster after upgrade

Example of upgrade:

```
---
- type: cluster
  name: control
  version: 2.4
  description: "Monitoring and Control Software"

  upgrade:
      -
          versions:
              min: 0.4
              max: 2.0
          description: New cool upgrade
          states:
              available: [any]
              on_success: upgradable
      -
          versions:
              min: 1.0
              max: 3.0
          description: Super new upgrade
          name: upgrade 2
          states:
              available: [created, installed]
              on_success: ver2.4
```

Глава 3

Ansible module: adcm_config

```
> FILE    (/root/.local/lib/python3.7/site-packages/ansible/plugins/lookup/adcm_config.py)

        This lookup set value of specified config key/subkey for host,
        cluster or service

OPTIONS (= is mandatory):

= _terms
        cluster|service|host, 'key/subkey', value

NOTES:
      * if you run service action, you don't need specify
        service name

AUTHOR: Konstantin Voschanov <vka@arenadata.io>

EXAMPLES:
- debug: msg="set host config {{lookup('adcm_config', 'host', 'ssh-key', 'F25') }}"

- debug: msg="set cluster config {{lookup('adcm_config', 'cluster', 'adh.cfg/port', 80) }}"

- debug: msg="set service config {{lookup('adcm_config', 'service', 'adh.cfg/port', 80) }}"

- debug: msg="set service config {{lookup('adcm_config', 'service', 'adh.cfg/port', 80, service_name=
↪'ZOOKEEPER') }}"

RETURN VALUES:

  _raw:
    description:
      - new value of config
```