Arenadata™ Cluster Manager

Версия - develop

Software Development Kit

2022 Arenadata™

Оглавление

1 Bundle **3**
- 1.1 Cluster Bundle . . . 3
- 1.2 Host Bundle . . . 3

2 Config File **4**
- 2.1 Prototype . . . 5
- 2.2 Actions . . . 7
- 2.3 Components . . . 12
- 2.4 Config Parameters . . . 14
- 2.5 Export . . . 21
- 2.6 Import . . . 22
- 2.7 Upgrade . . . 23
- 2.8 Config order convention . . . 24

3 ArenaData Cluster Manager API **26**
- 3.1 Intro . . . 26
- 3.2 API Sections . . . 26
- 3.3 adcm . . . 27
- 3.4 cluster . . . 29
- 3.5 component . . . 43
- 3.6 group . . . 45
- 3.7 host . . . 46
- 3.8 info . . . 48
- 3.9 job . . . 49
- 3.10 logout . . . 49
- 3.11 profile . . . 50
- 3.12 provider . . . 51
- 3.13 role . . . 56
- 3.14 service . . . 57
- 3.15 stack . . . 63
- 3.16 stats . . . 67
- 3.17 swagger . . . 67
- 3.18 task . . . 67
- 3.19 token . . . 68
- 3.20 user . . . 68

4 Getting Heart Beat information **71**
- 4.1 Heart Beat for Components . . . 71
- 4.2 Heart Beat for Service . . . 71

4.3 Heart Beat for Cluster . . . 72

5 Ansible Modules and Plugins **73**
5.1 Object Multi State Set (adcm_multi_state_set) . . . 73
5.2 Object Multi State Unset (adcm_multi_state_unset) . . . 74
5.3 Object State Set (adcm_state) . . . 76
5.4 Check Log Entry Add (adcm_check) . . . 77
5.5 Custome Log Add (adcm_custom_log) . . . 78
5.6 Config Set (adcm_config) . . . 78
5.7 Host Add (adcm_add_host) . . . 80
5.8 Host Delete (adcm_delete_host) . . . 80
5.9 Host Add to Cluster (adcm_remove_host_from_cluster) . . . 80
5.10 Host Remove from Cluster (adcm_add_host_to_cluster) . . . 81
5.11 Host Component Map Set (adcm_hc) . . . 81
5.12 Service Delete (adcm_delete_service) . . . 82

6 Adcm bundles packer **83**
6.1 Naming rules . . . 83
6.2 Spec files . . . 83

The document provides information for work with the ADCM platform. The guide can be useful to administrators, programmers, developers and employees of the information technology departments that support the platform.

Important: Support – e-mail: info@arenadata.io

Глава 1

Bundle

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

All directories and files relative 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:

```
POST /api/v1/stack/load/
```

When loaded, **ADCM** uncompress bundle-file, search for every file with name *config.yaml* and treat them as config_.

Bundle could be one of two types: Cluster Bundle or Host Bundle.

1.1 Cluster Bundle

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

1.2 Host Bundle

Host bundle should include exactly once definition of a host prototype and one or zero definition of a host provider prototype.

Глава 2

Config File

Config file is a way to describe how **ADCM** should run ansible playbooks and which parameters should be asked from End User.

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

That file has to be named as *config.yaml* or *config.yml*. Multiple *config.yaml* files are allowed in a bundle_.

- *Prototype*
- *Actions*
  - *Params*
  - *States*
  - *Masking*
  - *Config*
  - *Log Files*
  - *Allow to terminate*
  - *UI Options*
  - *Host action*
  - *Host Component Map ACL*
  - *Multi Scripts*
  - *Buttons*
  - *venv*
- *Components*
  - *Constraint*
  - *Requires*
- *Config Parameters*
  - *Activatable groups*

        - *Config customization*
        - *Common config parameters*
        - *Read Only / Writable*
        - *UI Options*
        - *Types of config parameters*
    - *Export*
    - *Import*
        - *Optional import parameters*
            * *Versions*
            * *Required*
            * *Default*
            * *Multibind*
    - *Upgrade*
        - *from_edition*
    - *Config order convention*

2.1 Prototype

Prototype is a description of one object.

Every prototype should be one of the following types:

- cluster;
- service;
- provider (host provider);
- host.

Every prototype has following mandatory properties:

- type;
- name;
- version.

So minimal working prototype should be like that:

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

Prototype definition can has same optional properties:

- Display Name and Description

```
---
- type: cluster
```

```
  name: control
  version: 1
  display_name: "Controlling Software"
  description: |
     This software is intended for monitoring and controlling
     you cluster and does it by ...

+ ``display_name`` is relatively short human readable name;

+ ``name`` is really short name intended for use in ansible scripts.
```

- Required

Every service prototype can have an optional `required` property. Default value is `false`. If cluster cannot work without this service, set it to `true`. When required service is not added to cluster, cluster will have an issue.

- Monitoring

Every service, component or host prototype can have an optional `monitoring` property. It defines, should their instances be monitored or not. Default value is `active`. If you want to exclude them from monitoring system, set it to `passive`.

- ADCM min version:

Any cluster or host provider prototype can have an optional `adcm_min_version` property. This field define minimal version of **ADCM** required for this bundle_ to work correctly. If installed **ADCM** instance has version less than `adcm_min_version` defined in bundle, loading of this bundle cause a error.

```
---
- type: cluster
  name: control
  version: 1
  adcm_min_version: 2019.05.01.00
```

- License

Every cluster or host provider prototype can have an optional `license` property (corresponding to the cluster or the host bundle_ respectively). The value of a `license` property is a path to the licence file relative to a bundle root.

If prototype has a `license` property, a user of **ADCM** will be asked to accept a license after the bundle is loaded. Before a user accept a license any action with the bundle will be blocked.

```
---
- type: cluster
  name: adh
  display_name: Arenadata Hadoop
  version: 1
  license: misc/license.txt
```

Note: You can use a special dot syntax to point out a path to a license file. In this case a license file will be searched in the same directory as *config.yaml*:

```
license: ./license.txt
```

- Edition

Every cluster or host provider prototype can have an optional `edition` property (corresponding the cluster or the host bundle_ respectively).

Default value of `edition` is `community`. Value of `edition` can be used in *Upgrade* definition.

```
---
- type: cluster
  name: adh
  display_name: Arenadata Hadoop
  version: 1
  edition: enterprise
```

2.2 Actions

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

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

  actions:
    install:
      display_name: "Install Monitor Server"
      description: |
        By click on this button you install monitoring and controlling server ...
      type: job
      script_type: ansible
      script: ansible/site.yaml
      allow_to_terminate: true
      params:
        ansible_tags: install
        jinja2_native: true
      states:
        available:
          - created
        on_success: installed
        on_fail: created
      config:
        quorum:
          type: integer
      log_files:
        - check
```

You can see one action called `install`. That action has a `job` **type** and its `script` property specify an ansible playbook that resides in directory *<bundle_root>/ansible/site.yaml*.

Note: You can use a special dot syntax to point out a path to a script file. In this case a script file will be searched in the same directory as *config.yaml*.

```
script: ./site.yaml
```

2.2.1 Params

You can 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 turns to be `--tags` parameter of ansible-playbook call.

You can also pass the *jinja2_native* parameter. This parameter will be written to *ansible.cfg* file. This option preserves variable types during template operations.

2.2.2 States

Action can have a `states` optional statement. It defines 3 clauses:

- `available` – list of states of object for which action will be available;
- `on_success` – state in which object will be set in case of action success;
- `on_fail` – state in which object will be set in case of action fail.

States `on_success` and `on_fail` are optional. If they are missed, state of an object will not be changed after action success or/and fail respectively.

2.2.3 Masking

Masking is a new DSL for state and multistate operation. This DSL is not compatible with States DSL provided above.

```
---
actions:
  install:
  display_name: "Install Monitor Server"
  type: job
  masking:
      # Action will be shown if both condition under state and multi_state are met.
      state:
          available:
              # If the state equal to any of this, then condition is true
              - "state_value1"
              - "state_value2"
          unavailable:
              # If you place unavailable, then no available should be there
              - "state_value1"
              - "state_value2"
      multi_state:
          available:
              # If we have any of this multistate, then condition is true
              - "multi_state_1"
              - "multi_state_2"
          unavailable:
              # If you place unavailable, then no available should be there
              - "multi_state_3"
              - "multi_state_4"
  on_fail:
      state: "new_sate_value"
      multi_state:
          set:
              - "multi_state3"
          unset:
              - "multi_state4"
```

```
  on_success:
      state: "new_sate_value"
      multi_state:
          set:
              - "multi_state3"
          unset:
              - "multi_state4"
```

To simplify the rule description, you could use scalar "any" which is select all possible values "any" scalar selector.

```
---
  masking:
  state:
      available: "any"
      unavailable: "any"
  multi_state:
      available: "any"
      unavailable: "any"
```

If masking state is omitted, than we think that action available on every state or multi_state.

So the following variants are equal:

```
---
  masking:
```

```
---
  masking:
      state:
```

```
---
  masking:
      state:
          available: "any"
```

The following variants are equal too:

```
---
  masking:
```

```
---
  masking:
      multi_state:
```

```
---
  masking:
      multi_state:
          available: "any"
```

2.2.4 Config

Action can have a `config` optional statement. It can define a list of values as in *Config Parameters*. 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 }}
```

2.2.5 Log Files

Action can have a `log_files` optional statement. It can define a list of log tags. At now, only one log tag `check` is supported. If you has `check` log tag, you can use *adcm_check* ansible module.

2.2.6 Allow to terminate

Action can have a `allow_to_terminate` optional statement. Boolean value. If value not explicitly set, default value would be `false`. If the value `allow_to_terminate` is `true`, then it will be possible to cancel the running ansible playbook, otherwise it will be impossible to cancel the playbook.

2.2.7 UI Options

That block allow you to tweak how UI show and perform an action.

Disclaimer: Ask user if he really want to run this action and show disclaimer text in a modal window.

```
actions:
  install:
    type: job
    script_type: ansible
    script: ansible/site.yaml
    ui_options:
       disclaimer: "Are you really want to click me?"
```

2.2.8 Host action

Action can have a `host_action` optional statement. Boolean value. If value not explicitly set, default value would be `false`. If the value is `true`, then the action will be performed on the selected host, and a *target* group will be created, which will include one selected host.

```
actions:
  install:
    type: job
    script_type: ansible
    script: action.yaml
    host_action: true
```

2.2.9 Host Component Map ACL

Action can have a `hc_acl` optional statement. If `hc_acl` is present, when action is run via UI, new host component map will be asked from the user before action start.

```
actions:
  expand:
    type: job
    script_type: ansible
    script: ansible/site.yaml
    states:
      available: all
    hc_acl:
      -
        service: hadoop
        component: datanode
```

```
      action: add
    -
      service: hadoop
      component: server
      action: remove
```

`hc_acl` define list of 3 clauses which specify permissions to modify component distribution by hosts:

- `service` – name of service to work with;
- `component` – name of component to work with;
- `action` – permission type. Can be `add` or `remove`.

2.2.10 Multi Scripts

Action can have more then one ansible playbook. In this case `type` of action should be `task`. Ansible scripts will be executed one by one in the order of description. If any of script fail, execution will be stopped and state of the object will be set to the value of `on_fail` property.

```
actions:
  install:
    type: task
    scripts:
      -
          name: prepare
          display_name: Prepare to install
          script_type: ansible
          script: ansible/prepare.yaml
          on_fail: created
      -
          name: install
          display_name: Actual install
          script_type: ansible
          script: ansible/install.yaml
          on_fail: prepare
    states:
      available:
        - created
        - prepare
      on_success: installed
```

2.2.11 Buttons

There is a way to bind an action to a button in UI. To do that check the following code:

```
actions:
  install:
    type: task
    scripts:
      -
          name: prepare
          display_name: Prepare to install
          script_type: ansible
          script: ansible/prepare.yaml
          on_fail: created
    button: "create_host"
```

Name	Description
create_host	“Create Host” button on Hosts page. If you bind multiple action to one button, than an action selection window appears

2.2.12 venv

There is a way to specify ansible version for `ansible` script type actions using `venv` optional statement.

Valid values:

- `default` – launch action in ansible 2.8 environment;
- `2.9` – launch action in ansible 2.9 environment;

By default this value is `default`

Change default environment for all actions on object:

```
---
- type: cluster # service component hostprovider host
  name: control
  version: 1

  venv: "2.9"
```

You also have a way to tweak this for any of actions:

```
actions:
  install:
    type: job
    script_type: ansible
    script: action.yaml
    venv: "2.9"
```

2.3 Components

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

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

  components:
    master:
      display_name: "Master Node"
      description: "This node control all data nodes (see below)"
      constraint: [1,2]
    node:
      display_name: "Data Node"
      constraint: [+]
    client:
      monitoring: passive
```

2.3.1 Constraint

Every component can has an optional `constraint` property. It describes how many instances of this component should be installed in one cluster:

- `[1]` – exactly once component should be installed;
- `[0,1]` – one or zero component should be installed;
- `[1,2]` – one or two component should be installed;
- `[0,+]` – zero or any more component should be installed (default value);
- `[1,odd]` – one or more component should be installed; total amount should be odd;
- `[0,odd]` – zero or more component should be installed; if more than zero, total amount should be odd;
- `[odd]` – the same as [1,odd];
- `[1,+]` – one or any more component should be installed;
- `[+]` – component should be installed on all hosts of cluster.

Every component can have an optional `monitoring` property. It defines, should component be monitored or not. Default value is `active`. If you want to exclude component from monitoring system, set it to `passive`.

2.3.2 Requires

Components can have an optional `requires` property, which describe dependencies between components, i.e. which components need another components to be installed in cluster simultaneously.

Each component can require one or more another component in the same service or in another service in this cluster. If component required another component, you could not add it to host component map (hc) when required component absent in hc.

Requires can be circular. In this case all required components should be installed simultaneously.

So, dependencies are another type of constraints (or meta constraints) for a component.

```
---
- type: service
  name: Hive
  version: 1.0

  components:
    HiveServer:
      requires:
        - component: Metastore
    Metastore:
      constraint: [0,1]
    TezUI:
      requires:
        - service: YARN
          component: TimeLineServer
      constraint: [0,1]

- type: service
  name: YARN
  version: 2.0

  components:
    TimeLineServer:
      requires:
```

```
        - service: Hive
          component: TezUI
```

2.4 Config Parameters

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:
    - name: repos
      type: group
      subs:
         - name: ads
           display_name: "ADS repository URL"
           type: string
           default: https://ci.arenadata.io/artifactory/list/ads-centos7-x64-ads/
           required: no
```

In the example you can see parameter `ads` in group `repos`. That parameter is not required. This means that it is possible for the user to leave it empty.

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

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

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

Group is just a visual representation in UI.

There is an old, deprecated way to define groups as map of map, not array of map:

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

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

2.4.1 Activatable groups

Group can be activatable. This mean you can switch on or off this group via UI. When group is switched off (`active = false`) you don't need to fill in its sub values in config UI and it value in ansible inventory file would be null.

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

  config:
     - name: grafana
       type: group
       activatable: true
       active: false
       subs:
          - name: endpoint
            type: string
          - name: port
            type: integer
```

2.4.2 Config customization

There is a functionality in ADCM which allow End User to customize some paremeters of cluster (or service, or component, or provider) on some group of hosts. This functionality is disabled by default. To start working with it you have to option.

Change default policy for all config on object:

```
---
- type: cluster # service component hostprovider
  name: control
  version: 1

config_group_customization: true
```

By default this value is False

You also have a way to tweak this for any of parameters:

```
---
 - name: ads
   type: string
   group_customization: true/false
```

2.4.3 Common config parameters

Name	Mandatory	Values	Default
type	yes	see below	
display_name	no		
description	no		
default	no	user defined	per type
required	no	yes/no	yes
read_only	no	see below	no
writable	no	see below	yes
ui_options	no	see below	no

If `required` set to `yes`, it is required to have a non empty and valid value for the variable.

2.4.4 Read Only / Writable

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

Examples:

Parameter quorum will be read only (protected from change) when service state is `installed` or `running`:

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

Parameter quorum will be read only (protected from change) for all service states except `created`:

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

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

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

2.4.5 UI Options

That block allow you to tweak how UI draw config.

We have the following options:

- Invisible

Add "invisible: true" to your config to hide config from UI.

NOTE: That option doesn't hide information from API, and doesn't disable ReadOnly/Writable logic. It's just an UI tweak.

```
int:
  type: integer
  default: 1
  required: no
  ui_options:
    invisible: true
```

- No password confirmation

Sometimes you need to make a password field without confirmation (just one input). It especially useful for action's parameters.

```
passwd_no_confirm:
  type: password
  required: yes
```

```
  ui_options:
    no_confirm: true
```

- Advanced

There is a new checkbox on config view that allow you to hide some of the parameter which is marked as advanced.

NOTE: Parameter marked as advanced is not searchable while they are not visible (when Advanced check box is unchecked).

```
float:
  type: float
  default: 1.0
  required: no
  ui_options:
    advanced: true
```

2.4.6 Types of config parameters

- String – arbitrary string. The size is not limited. If default is not explicitly set, default value would be an empty string .
- Text – the same as string, but will be displayed as multi line text in user interface.
- Boolean – boolean value: `True` or `False`.
- Integer – integer number. If default is 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 is 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 displays 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
```

- List – dynamic array of arbitrary length. Array elements should be a string. Any element can be added, deleted or modified on every config update:

```yaml
config:
  mount_points:
    type: list
    default:
      - /dev/rdisk0s1
      - /dev/rdisk0s2
      - /dev/rdisk0s3
```

- Map – dynamic key-value dictionary of arbitrary length. Keys and values should be a string. Any key or/and value can be added, deleted or modified on every config update:

```yaml
config:
  person:
    type: map
    default:
      name: Joe
      age: "24"
      sex: m
```

- Variant – variant is a type where user can select one value from a list of options in UI. Sources of variants can be different. You can get a list of variants from another config parameter (type of this config parameter should be a list):

```yaml
config:
  - name: mount
    type: variant
    source:
      type: config
      name: mount_points

  - name: mount_points
    type: list
    default:
      - /dev/rdisk0s1
      - /dev/rdisk0s2
      - /dev/rdisk0s3
    read_only: any
    ui_options:
      invisible: true
```

Also you can get a list of variants from builtin **ADCM** functions, such as `cluster_hosts`, which return list of all hosts in current cluster:

```yaml
config:
  - name: cluster_host
    type: variant
    source:
      type: builtin
      name: host_in_cluster
```

Builtin **ADCM** functions:

- `host_in_cluster` – return list of all hosts in current cluster;
- `host_not_in_clusters` – return list of all hosts not included in any cluster;
- `service_in_cluster` – return list of all services installed in current cluster;
- `service_to_add` – return list of all services not installed in current cluster;

- `host` – see below.
- Host – builtin function `host` can receive a lot of arguments in form of `predicate` and `args`: `predicate` is a name of internal host function; `args` are arguments for this function. All internal functions return list of hosts (of course, this list can be empty). For example:

```
config:
  - name: vhost
    type: variant
    source:
       type: builtin
       name: host
       args:
         predicate: and
         args:
           - predicate: in_service
             args:
               service: UBER
           - predicate: or
             args:
               - predicate: in_component
                 args:
                   service: UBER
                   component: UBER_SERVER
               - predicate: in_cluster
                 args:
```

List of internal host functions:

- `in_cluster` – return list of all hosts in current cluster. Args: None.
- `in_service` – return list of all hosts in specified service. Args: service name.
- `not_in_service` – return list of all hosts not in specified service. Args: service name.
- `in_component` – return list of all hosts in specified component. Args: service and components names.
- `not_in_component` – return list of all hosts that are not in specified component. Args: service and components names.
- `in_hc` – return list of all hosts in host component map of this cluster. Args: None.
- `not_in_hc` – return list of all hosts that are not in host component map of this cluster. Args: None.
- `and` – return union of sets of its args.
- `or` – return intersection of sets of its args.

Or you can list variants right here in a bundle file:

```
config:
  - name: city
    type: variant
    source:
      type: inline
      strict: False
      value:
        - Moscow
        - Voronez
        - Ufa
    default: Samara
```

You can also specify optional argument `strict` for variants source. If `strict` is set to `False`, user can not only select value from predefined list, but also enter his own values in UI.

- JSON – this type can store arbitrary json-data. JSON structure is not preserved and can be modified on every config update:

```
config:
  population:
    type: json
    default:
      -
        country: Russia
        cities:
          -
             name: Moscow
             population: 9
      -
        country: USA
        cities:
          -
             name: LA
             population: 5
```

- Structure – this type can store some regular data described by a *yspec*-specification. On every config update new data are checked against specification. If input data and specification do not match, ADCM will return a error.

  Documentation on yspec is available at https://pypi.org/project/yspec/.

```
config:
  country_codes:
    type: structure
    yspec: codes/schema.yaml
    default:
      -
         country: Greece
         code: 30
      -
         country: France
         code: 33
      -
         country: Spain
         code: 34
```

- schema.yaml file:

```
---
root:
  match: list
  item: country_code

country_code:
  match: dict
  items:
    country: string
    code: integer

string:
  match: string
```

```
integer:
  match: int
```

Note: You can use a special dot syntax to point out a path to a *yspec*-specification file. In this case a specification file will be searched in the same directory as *config.yaml*:

```
yspec: ./schema.yaml
```

- File – the content of this property is stored as a file on a file system. When you reference this variable in an ansible script it returns a full path to this file.

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

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

It can be usable to store ssh private keys for an ansible script:

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

  config:
    ansible_private_key_file:
      type: file
      default: host/ssh.key
```

Note: You can use a special dot syntax to point out a path to a default file. In this case a default file will be searched in the same directory as *config.yaml*:

```
config:
  ansible_private_key_file:
    type: file
    default: ./ssh.key
```

2.5 Export

If you want to share some parts of config with other clusters, you can use `export` statement. In `export` you can specify any number of the 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: integer
     quorum:
        type: integer
        default: 3

  export:
```

```
      - core-site
      - quorum
```

2.6 Import

To use exported parts of config from other cluster you should use `import` statement in your cluster. In `import` you should specify name. If you don't need any optional parameters use empty dict.

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

  import:
     Hadoop:
        versions:
           min: 1.8
           max_strict: 2.5

     grafana: {}
```

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

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

2.6.1 Optional import parameters

Versions

You could specify the interval of versions, appropriate for import. Min and max use non strict comparison (i.e. boundaries include version of cluster/host). If you want strict comparison, you can use min_strict and/or max_strict instead of min/max. If you won't specify borders appropriate version wouldn't check.

Required

Import statement can have optional `required` parameter. If `required` is set to `true`, this cluster or service will raise issue and block any actions before required import is properly binded with export cluster/service. Default value for `required` is `false`.

Default

Import statement can have optional `default` parameter:

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

  config:
     endpoints:
        host:
```

```
          type: string
       port:
          type: integer
          default: 80

  import:
     Graphite:
        versions:
           min: 1.8
           max: 2.5
        default:
           - endpoints
```

`default` parameter should be an array of names of config group of this cluster or service. If this cluster/service is not binded with any export cluster/service import section in ansible inventory file will be filled with values from specified config group.

To your own safety name of default groups should be the same as export group names.

Multibind

Import statement can have optional `multibind` parameter. If it set to `true`, you can bind many instances of cluster or service to the same cluster/service. In ansible inventory file this import values will be represented as an array.

Default value of `multibind` is `false`.

2.7 Upgrade

Cluster or host prototype can have an `upgrade` statement. This statement describes how to upgrade from one version of bundle_ to another. Prototype can have a few upgrade items to describe upgrade from different versions. Each item should include:

- `versions` – interval of versions, appropriate for upgrade;
- `name`;
- `description` (optional);
- `states`:
- `available` – list states of cluster, permitted to upgrade. Special value `any` can be used for upgrade, which available on any state;
- `on_success` – state of cluster after upgrade.

Example of upgrade:

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

  upgrade:
     -
        name: Upgrade 1
        versions:
           min: 0.4
```

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

About versions comparison: `min` and `max` use non strict comparison (i.e. boundaries include version of cluster/host). If you want strict comparison, you can use `min_strict` and/or `max_strict` instead of `min`/`max`.

2.7.1 from_edition

Upgrade can has optional `from_edition` property. The value of `from_edition` property is list of editions from which upgrades are permitted.

Special value `any` can be used to specify permission to upgrade from bundles with any edition.

Default value of `from_edition` is `[community]`.

```
upgrade:
   -
      name: Upgrade 1
      versions:
         min: 0.4
         max: 2.0
      description: New cool upgrade
      states:
         available: [created, installed]
         on_success: upgradable
      from_edition:
         - community
         - enterprise
```

2.8 Config order convention

To comply with the uniformity and transparency for configuration, it is desirable to follow the order of the prototypes. It is recommended to follow the order as shown in the example:

```
---
- name:
  type: cluster
  description:
  version: &version

  upgrade:
    ...

  config:
```

```
    ...

  import:
    ...

  actions:

    install:
      ...

- type: service
  name:
  version:
  description:

  config:
    ...

  components:
    ...

  actions:

    install:
      ...
```

Глава 3

ArenaData Cluster Manager API

Version: 1.0

3.1 Intro

Cluster: Set of hosts and services

Host: Abstract container for components

Service: Set of components

Component: Piece of software

HostComponent: Map between hosts and components of services

Task: Service action at work. Consists of one or more jobs

Job: Minimal quantum of action. Run ansible playbook to do actual work

3.2 API Sections

adcm

cluster

component

group

host

info

job

logout

profile

provider

role

service

stack

stats

swagger

task

token

user

3.3 adcm

3.3.1 list

GET `/api/v1/adcm/`

List adcm object

Query Parameters

3.3.2 read

GET `/api/v1/adcm/{adcm_id}/`

Show adcm object

Path Parameters

The following parameters should be included in the URL path.

3.3.3 action > list

GET `/api/v1/adcm/{adcm_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.3.4 action > read

GET `/api/v1/adcm/{adcm_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.3.5 action > run > create

POST `/api/v1/adcm/{adcm_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.3.6 config > list

GET `/api/v1/adcm/{adcm_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.3.7 config > current > list

GET `/api/v1/adcm/{adcm_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.3.8 config > history > list

GET `/api/v1/adcm/{adcm_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.3.9 config > history > create

POST `/api/v1/adcm/{adcm_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.3.10 config > history > read

GET `/api/v1/adcm/{adcm_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.3.11 config > history > restore > partial_update

PATCH `/api/v1/adcm/{adcm_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.3.12 config > previous > list

GET `/api/v1/adcm/{adcm_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4 cluster

3.4.1 list

GET `/api/v1/cluster/`

List of all existing clusters

Query Parameters

3.4.2 create

POST `/api/v1/cluster/`

Create new cluster

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.3 read

GET `/api/v1/cluster/{cluster_id}/`

Show cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.4 partial_update

PATCH `/api/v1/cluster/{cluster_id}/`

Edit cluster

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.5 delete

DELETE `/api/v1/cluster/{cluster_id}/`

Remove cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.6 action > list

GET `/api/v1/cluster/{cluster_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.7 action > read

GET `/api/v1/cluster/{cluster_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.4.8 action > run > create

POST `/api/v1/cluster/{cluster_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.9 bind > list

GET `/api/v1/cluster/{cluster_id}/bind/`

List all binds of specified cluster

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.10 bind > create

POST `/api/v1/cluster/{cluster_id}/bind/`

Bind two clusters

Path Parameters

The following parameters should be included in the URL path.

3.4.11 bind > read

GET `/api/v1/cluster/{cluster_id}/bind/{bind_id}/`

Show specified bind of specified cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.12 bind > delete

DELETE `/api/v1/cluster/{cluster_id}/bind/{bind_id}/`

Unbind specified bind of specified cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.13 config > list

GET `/api/v1/cluster/{cluster_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.14 config > current > list

GET `/api/v1/cluster/{cluster_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.15 config > history > list

GET `/api/v1/cluster/{cluster_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.16 config > history > create

POST `/api/v1/cluster/{cluster_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.17 config > history > read

GET `/api/v1/cluster/{cluster_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.4.18 config > history > restore > partial_update

PATCH `/api/v1/cluster/{cluster_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.19 config > previous > list

GET `/api/v1/cluster/{cluster_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.20 host > list

GET `/api/v1/cluster/{cluster_id}/host/`

List all hosts

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.21 host > create

POST `/api/v1/cluster/{cluster_id}/host/`

Create host

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.22 host > read

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/`

Show host

Path Parameters

The following parameters should be included in the URL path.

3.4.23 host > delete

DELETE `/api/v1/cluster/{cluster_id}/host/{host_id}/`

Delete host

Path Parameters

The following parameters should be included in the URL path.

3.4.24 host > action > list

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.25 host > action > read

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.4.26 host > action > run > create

POST `/api/v1/cluster/{cluster_id}/host/{host_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.27 host > config > list

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.28 host > config > current > list

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.29 host > config > history > list

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.30 host > config > history > create

POST `/api/v1/cluster/{cluster_id}/host/{host_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.31 host > config > history > read

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.4.32 host > config > history > restore > partial_update

PATCH `/api/v1/cluster/{cluster_id}/host/{host_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.33 host > config > previous > list

GET `/api/v1/cluster/{cluster_id}/host/{host_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.34 hostcomponent > list

GET `/api/v1/cluster/{cluster_id}/hostcomponent/`

Show host <-> component map in a specified cluster

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.35 hostcomponent > create

POST `/api/v1/cluster/{cluster_id}/hostcomponent/`

Create new mapping service:component <-> host in a specified cluster.

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.36 hostcomponent > read

GET `/api/v1/cluster/{cluster_id}/hostcomponent/{hs_id}/`

Show host <-> component link in a specified cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.37 import > list

GET `/api/v1/cluster/{cluster_id}/import/`

List all imports available for specified cluster

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.38 import > create

POST `/api/v1/cluster/{cluster_id}/import/`

Update bind for cluster

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.39 service > list

GET `/api/v1/cluster/{cluster_id}/service/`

List all services

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.40 service > create

POST `/api/v1/cluster/{cluster_id}/service/`

Add service to cluster

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.41 service > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/`

Show service

Path Parameters

The following parameters should be included in the URL path.

3.4.42 service > delete

DELETE `/api/v1/cluster/{cluster_id}/service/{service_id}/`

Remove service from cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.43 service > action > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.44 service > action > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.4.45 service > action > run > create

POST `/api/v1/cluster/{cluster_id}/service/{service_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.46 service > bind > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/bind/`

List all binds of service

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.47 service > bind > create

POST `/api/v1/cluster/{cluster_id}/service/{service_id}/bind/`

Bind two services

Path Parameters

The following parameters should be included in the URL path.

3.4.48 service > bind > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/bind/{bind_id}/`

Show specified bind of service

Path Parameters

The following parameters should be included in the URL path.

3.4.49 service > bind > delete

DELETE `/api/v1/cluster/{cluster_id}/service/{service_id}/bind/{bind_id}/`

Unbind specified bind of service

Path Parameters

The following parameters should be included in the URL path.

3.4.50 service > component > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/`

List all components

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.51 service > component > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/`

Show component

Path Parameters

The following parameters should be included in the URL path.

3.4.52 service > component > action > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.53 service > component > action > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.4.54 service > component > action > run > create

POST `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/action/{action_id}/ru`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.55 service > component > config > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.56 service > component > config > current > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.57 service > component > config > history > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.58 service > component > config > history > create

POST `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.59 service > component > config > history > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/history/{versio`

Path Parameters

The following parameters should be included in the URL path.

3.4.60 service > component > config > history > restore > partial_update

PATCH `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/history/{ver`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.61 service > component > config > previous > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.62 service > config > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.63 service > config > current > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.64 service > config > history > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.65 service > config > history > create

POST `/api/v1/cluster/{cluster_id}/service/{service_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.66 service > config > history > read

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.4.67 service > config > history > restore > partial_update

PATCH `/api/v1/cluster/{cluster_id}/service/{service_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.68 service > config > previous > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.69 service > import > list

GET `/api/v1/cluster/{cluster_id}/service/{service_id}/import/`

List all imports available for specified service

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.70 service > import > create

POST `/api/v1/cluster/{cluster_id}/service/{service_id}/import/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.4.71 serviceprototype > list

GET `/api/v1/cluster/{cluster_id}/serviceprototype/`

List all services of specified cluster of bundle

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.72 status > list

GET `/api/v1/cluster/{cluster_id}/status/`

Show all hosts and components in a specified cluster

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.73 upgrade > list

GET `/api/v1/cluster/{cluster_id}/upgrade/`

List all available upgrades for specified cluster

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.4.74 upgrade > read

GET `/api/v1/cluster/{cluster_id}/upgrade/{upgrade_id}/`

List all available upgrades for specified cluster

Path Parameters

The following parameters should be included in the URL path.

3.4.75 upgrade > do > create

POST `/api/v1/cluster/{cluster_id}/upgrade/{upgrade_id}/do/`

Do upgrade specified cluster

Path Parameters

The following parameters should be included in the URL path.

3.5 component

3.5.1 list

GET `/api/v1/component/`

List all components

Query Parameters

3.5.2 read

GET `/api/v1/component/{component_id}/`

Show component

Path Parameters

The following parameters should be included in the URL path.

3.5.3 action > list

GET `/api/v1/component/{component_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.5.4 action > read

GET `/api/v1/component/{component_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.5.5 action > run > create

POST `/api/v1/component/{component_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.5.6 config > list

GET `/api/v1/component/{component_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.5.7 config > current > list

GET `/api/v1/component/{component_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.5.8 config > history > list

GET `/api/v1/component/{component_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.5.9 config > history > create

POST `/api/v1/component/{component_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.5.10 config > history > read

GET `/api/v1/component/{component_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.5.11 config > history > restore > partial_update

PATCH `/api/v1/component/{component_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.5.12 config > previous > list

GET `/api/v1/component/{component_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.6 group

3.6.1 list

GET `/api/v1/group/`

List all existing user groups

Query Parameters

3.6.2 create

POST `/api/v1/group/`

Create new user group

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.6.3 read

GET `/api/v1/group/{name}/`

show user group

Path Parameters

The following parameters should be included in the URL path.

3.6.4 delete

DELETE `/api/v1/group/{name}/`

delete user group

Path Parameters

The following parameters should be included in the URL path.

3.6.5 role > create

POST `/api/v1/group/{name}/role/`

Add group role

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.6.6 role > delete

DELETE `/api/v1/group/{name}/role/`

Remove group role

Path Parameters

The following parameters should be included in the URL path.

3.7 host

3.7.1 list

GET `/api/v1/host/`

List all hosts

Query Parameters

3.7.2 create

POST `/api/v1/host/`

Create host

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.7.3 read

GET `/api/v1/host/{host_id}/`

Show host

Path Parameters

The following parameters should be included in the URL path.

3.7.4 delete

DELETE `/api/v1/host/{host_id}/`

Delete host

Path Parameters

The following parameters should be included in the URL path.

3.7.5 action > list

GET `/api/v1/host/{host_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.7.6 action > read

GET `/api/v1/host/{host_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.7.7 action > run > create

POST `/api/v1/host/{host_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.7.8 config > list

GET `/api/v1/host/{host_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.7.9 config > current > list

GET `/api/v1/host/{host_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.7.10 config > history > list

GET `/api/v1/host/{host_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.7.11 config > history > create

POST `/api/v1/host/{host_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.7.12 config > history > read

GET `/api/v1/host/{host_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.7.13 config > history > restore > partial_update

PATCH `/api/v1/host/{host_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.7.14 config > previous > list

GET `/api/v1/host/{host_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.8 info

3.8.1 list

GET `/api/v1/info/`

General info about ADCM

Query Parameters

3.9 job

3.9.1 list

GET `/api/v1/job/`

List all jobs

Query Parameters

3.9.2 read

GET `/api/v1/job/{job_id}/`

Show job

Path Parameters

The following parameters should be included in the URL path.

3.9.3 log > list

GET `/api/v1/job/{job_id}/log/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.9.4 log > read

GET `/api/v1/job/{job_id}/log/{log_id}/`

Path Parameters

The following parameters should be included in the URL path.

3.9.5 log > read_0

GET `/api/v1/job/{job_id}/log/{tag}/{level}/{log_type}/`

Show log file

Path Parameters

The following parameters should be included in the URL path.

3.10 logout

3.10.1 create

POST `/api/v1/logout/`

Logout user from Django session

3.11 profile

3.11.1 list

GET `/api/v1/profile/`

List all existing user's profiles

Query Parameters

3.11.2 create

POST `/api/v1/profile/`

Create new user profile

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.11.3 read

GET `/api/v1/profile/{username}/`

Show user profile

Path Parameters

The following parameters should be included in the URL path.

3.11.4 partial_update

PATCH `/api/v1/profile/{username}/`

Edit user profile

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.11.5 delete

DELETE `/api/v1/profile/{username}/`

delete user and profile

Path Parameters

The following parameters should be included in the URL path.

3.11.6 password > partial_update

PATCH `/api/v1/profile/{username}/password/`

Change user password

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12 provider

3.12.1 list

GET `/api/v1/provider/`

List all host providers

Query Parameters

3.12.2 create

POST `/api/v1/provider/`

Create new host provider

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.3 read

GET `/api/v1/provider/{provider_id}/`

Show host provider

Path Parameters

The following parameters should be included in the URL path.

3.12.4 delete

DELETE `/api/v1/provider/{provider_id}/`

Remove host provider

Path Parameters

The following parameters should be included in the URL path.

3.12.5 action > list

GET `/api/v1/provider/{provider_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.6 action > read

GET `/api/v1/provider/{provider_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.12.7 action > run > create

POST `/api/v1/provider/{provider_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.8 config > list

GET `/api/v1/provider/{provider_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.9 config > current > list

GET `/api/v1/provider/{provider_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.10 config > history > list

GET `/api/v1/provider/{provider_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.11 config > history > create

POST `/api/v1/provider/{provider_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.12 config > history > read

GET `/api/v1/provider/{provider_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.12.13 config > history > restore > partial_update

PATCH `/api/v1/provider/{provider_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.14 config > previous > list

GET `/api/v1/provider/{provider_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.15 host > list

GET `/api/v1/provider/{provider_id}/host/`

List all hosts

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.16 host > create

POST `/api/v1/provider/{provider_id}/host/`

Create host

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.17 host > read

GET `/api/v1/provider/{provider_id}/host/{host_id}/`

Show host

Path Parameters

The following parameters should be included in the URL path.

3.12.18 host > delete

DELETE `/api/v1/provider/{provider_id}/host/{host_id}/`

Delete host

Path Parameters

The following parameters should be included in the URL path.

3.12.19 host > action > list

GET `/api/v1/provider/{provider_id}/host/{host_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.20 host > action > read

GET `/api/v1/provider/{provider_id}/host/{host_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.12.21 host > action > run > create

POST `/api/v1/provider/{provider_id}/host/{host_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.22 host > config > list

GET `/api/v1/provider/{provider_id}/host/{host_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.23 host > config > current > list

GET `/api/v1/provider/{provider_id}/host/{host_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.24 host > config > history > list

GET `/api/v1/provider/{provider_id}/host/{host_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.25 host > config > history > create

POST `/api/v1/provider/{provider_id}/host/{host_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.26 host > config > history > read

GET `/api/v1/provider/{provider_id}/host/{host_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.12.27 host > config > history > restore > partial_update

PATCH `/api/v1/provider/{provider_id}/host/{host_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.12.28 host > config > previous > list

GET `/api/v1/provider/{provider_id}/host/{host_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.29 upgrade > list

GET `/api/v1/provider/{provider_id}/upgrade/`

List all available upgrades for specified host provider

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.12.30 upgrade > read

GET `/api/v1/provider/{provider_id}/upgrade/{upgrade_id}/`

List all available upgrades for specified host provider

Path Parameters

The following parameters should be included in the URL path.

3.12.31 upgrade > do > create

POST `/api/v1/provider/{provider_id}/upgrade/{upgrade_id}/do/`

Do upgrade specified host provider

Path Parameters

The following parameters should be included in the URL path.

3.13 role

3.13.1 list

GET `/api/v1/role/`

List all existing roles

Query Parameters

3.13.2 read

GET `/api/v1/role/{role_id}/`

show role

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14 service

3.14.1 list

GET `/api/v1/service/`

List all services

Query Parameters

3.14.2 create

POST `/api/v1/service/`

Add service to cluster

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.3 read

GET `/api/v1/service/{service_id}/`

Show service

Path Parameters

The following parameters should be included in the URL path.

3.14.4 delete

DELETE `/api/v1/service/{service_id}/`

Remove service from cluster

Path Parameters

The following parameters should be included in the URL path.

3.14.5 action > list

GET `/api/v1/service/{service_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.6 action > read

GET `/api/v1/service/{service_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.14.7 action > run > create

POST `/api/v1/service/{service_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.8 bind > list

GET `/api/v1/service/{service_id}/bind/`

List all binds of service

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.9 bind > create

POST `/api/v1/service/{service_id}/bind/`

Bind two services

Path Parameters

The following parameters should be included in the URL path.

3.14.10 bind > read

GET `/api/v1/service/{service_id}/bind/{bind_id}/`

Show specified bind of service

Path Parameters

The following parameters should be included in the URL path.

3.14.11 bind > delete

DELETE `/api/v1/service/{service_id}/bind/{bind_id}/`

Unbind specified bind of service

Path Parameters

The following parameters should be included in the URL path.

3.14.12 component > list

GET `/api/v1/service/{service_id}/component/`

List all components

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.13 component > read

GET `/api/v1/service/{service_id}/component/{component_id}/`

Show component

Path Parameters

The following parameters should be included in the URL path.

3.14.14 component > action > list

GET `/api/v1/service/{service_id}/component/{component_id}/action/`

List all actions of a specified object

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.15 component > action > read

GET `/api/v1/service/{service_id}/component/{component_id}/action/{action_id}/`

Show specified action

Path Parameters

The following parameters should be included in the URL path.

3.14.16 component > action > run > create

POST `/api/v1/service/{service_id}/component/{component_id}/action/{action_id}/run/`

Ran specified action

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.17 component > config > list

GET `/api/v1/service/{service_id}/component/{component_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.18 component > config > current > list

GET `/api/v1/service/{service_id}/component/{component_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.19 component > config > history > list

GET `/api/v1/service/{service_id}/component/{component_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.20 component > config > history > create

POST `/api/v1/service/{service_id}/component/{component_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.21 component > config > history > read

GET `/api/v1/service/{service_id}/component/{component_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.14.22 component > config > history > restore > partial_update

PATCH `/api/v1/service/{service_id}/component/{component_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.23 component > config > previous > list

GET `/api/v1/service/{service_id}/component/{component_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.24 config > list

GET `/api/v1/service/{service_id}/config/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.25 config > current > list

GET `/api/v1/service/{service_id}/config/current/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.26 config > history > list

GET `/api/v1/service/{service_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.27 config > history > create

POST `/api/v1/service/{service_id}/config/history/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.28 config > history > read

GET `/api/v1/service/{service_id}/config/history/{version}/`

Path Parameters

The following parameters should be included in the URL path.

3.14.29 config > history > restore > partial_update

PATCH `/api/v1/service/{service_id}/config/history/{version}/restore/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.14.30 config > previous > list

GET `/api/v1/service/{service_id}/config/previous/`

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.31 import > list

GET `/api/v1/service/{service_id}/import/`

List all imports available for specified service

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.14.32 import > create

POST `/api/v1/service/{service_id}/import/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.15 stack

3.15.1 list

GET `/api/v1/stack/`

Operations with stack of services

Query Parameters

3.15.2 action > read

GET `/api/v1/stack/action/{action_id}/`

Show action

Path Parameters

The following parameters should be included in the URL path.

3.15.3 adcm > list

GET `/api/v1/stack/adcm/`

List adcm root object prototypes

Query Parameters

3.15.4 adcm > read

GET `/api/v1/stack/adcm/{prototype_id}/`

Show adcm prototype

Path Parameters

The following parameters should be included in the URL path.

3.15.5 bundle > list

GET `/api/v1/stack/bundle/`

List all bundles

Query Parameters

3.15.6 bundle > read

GET `/api/v1/stack/bundle/{bundle_id}/`

Show bundle

Path Parameters

The following parameters should be included in the URL path.

3.15.7 bundle > delete

DELETE `/api/v1/stack/bundle/{bundle_id}/`

Remove bundle

Path Parameters

The following parameters should be included in the URL path.

3.15.8 bundle > license > read

GET `/api/v1/stack/bundle/{bundle_id}/license/`

Path Parameters

The following parameters should be included in the URL path.

3.15.9 bundle > license > accept > update

PUT `/api/v1/stack/bundle/{bundle_id}/license/accept/`

Path Parameters

The following parameters should be included in the URL path.

3.15.10 bundle > update > update

PUT `/api/v1/stack/bundle/{bundle_id}/update/`

update bundle

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.15.11 cluster > list

GET `/api/v1/stack/cluster/`

List all cluster types

Query Parameters

3.15.12 cluster > read

GET `/api/v1/stack/cluster/{prototype_id}/`

Show cluster prototype

Path Parameters

The following parameters should be included in the URL path.

3.15.13 component > list

GET `/api/v1/stack/component/`

List all stack components

Query Parameters

3.15.14 component > read

GET `/api/v1/stack/component/{prototype_id}/`

Show component prototype

Path Parameters

The following parameters should be included in the URL path.

3.15.15 host > list

GET `/api/v1/stack/host/`

List all host types

Query Parameters

3.15.16 host > read

GET `/api/v1/stack/host/{prototype_id}/`

Show host prototype

Path Parameters

The following parameters should be included in the URL path.

3.15.17 load > create

POST `/api/v1/stack/load/`

Load bundle

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.15.18 load > servicemap > update

PUT `/api/v1/stack/load/servicemap/`

3.15.19 prototype > list

GET `/api/v1/stack/prototype/`

List all stack prototypes

Query Parameters

3.15.20 prototype > read

GET `/api/v1/stack/prototype/{prototype_id}/`

Show prototype

Path Parameters

The following parameters should be included in the URL path.

3.15.21 provider > list

GET `/api/v1/stack/provider/`

List all host providers types

Query Parameters

3.15.22 provider > read

GET `/api/v1/stack/provider/{prototype_id}/`

Show host provider prototype

Path Parameters

The following parameters should be included in the URL path.

3.15.23 service > list

GET `/api/v1/stack/service/`

List all stack services

Query Parameters

3.15.24 service > read

GET `/api/v1/stack/service/{prototype_id}/`

Show stack service

Path Parameters

The following parameters should be included in the URL path.

3.15.25 service > action > list

GET `/api/v1/stack/service/{prototype_id}/action/`

List all actions of a specified service

Path Parameters

The following parameters should be included in the URL path.

Query Parameters

3.15.26 upload > create

POST `/api/v1/stack/upload/`

Request Body

The request body should be a "**multipart/form-data**" encoded object, containing the following items.

3.16 stats

3.16.1 list

GET `/api/v1/stats/`

Statistics

Query Parameters

3.16.2 job > read

GET `/api/v1/stats/job/{job_id}/`

Show jobs stats

Path Parameters

The following parameters should be included in the URL path.

3.16.3 task > read

GET `/api/v1/stats/task/{task_id}/`

Show tasks stats

Path Parameters

The following parameters should be included in the URL path.

3.17 swagger

3.17.1 list

GET `/api/v1/swagger/`

3.18 task

3.18.1 list

GET `/api/v1/task/`

List all tasks

Query Parameters

3.18.2 create

POST `/api/v1/task/`

Create and run new task Return handler to new task

3.18.3 read

GET `/api/v1/task/{task_id}/`

Show task

Path Parameters

The following parameters should be included in the URL path.

3.18.4 cancel > update

PUT `/api/v1/task/{task_id}/cancel/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.18.5 restart > update

PUT `/api/v1/task/{task_id}/restart/`

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.19 token

3.19.1 create

POST `/api/v1/token/`

Provide authentication token

HTTP header for authorization:

`Authorization: Token XXXXX`

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.20 user

3.20.1 list

GET `/api/v1/user/`

List all existing users

Query Parameters

3.20.2 create

POST `/api/v1/user/`

Create new user

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.20.3 read

GET `/api/v1/user/{username}/`

show user

Path Parameters

The following parameters should be included in the URL path.

3.20.4 delete

DELETE `/api/v1/user/{username}/`

delete user and profile

Path Parameters

The following parameters should be included in the URL path.

3.20.5 group > create

POST `/api/v1/user/{username}/group/`

Add user to group

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.20.6 group > delete

DELETE `/api/v1/user/{username}/group/`

Remove user from group

Path Parameters

The following parameters should be included in the URL path.

3.20.7 password > partial_update

PATCH `/api/v1/user/{username}/password/`

Change user password

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.20.8 role > create

POST `/api/v1/user/{username}/role/`

Add user role

Path Parameters

The following parameters should be included in the URL path.

Request Body

The request body should be a "**application/json**" encoded object, containing the following items.

3.20.9 role > delete

DELETE `/api/v1/user/{username}/role/`

Remove user role

Path Parameters

The following parameters should be included in the URL path.

Глава 4

Getting Heart Beat information

For every object in **GET** `/api/v1/object/{object_id}/` you could get information about status of the components on hosts.

This information is just a Heart Beat of systemd-service or docker-container running in cluster.

4.1 Heart Beat for Components

GET `/status/api/v1/cluster/{cluster_id}/service/{service_id}/component/{component_id}/`

Parameter	Type
`status`	integer

Possible values of the `status`:

Value	Meaning
0	Service or container is up and running according to OS information on all nodes
<any>	Service or container is not up and running according to OS information on some of the nodes

4.2 Heart Beat for Service

Status for service is an aggregation of component's status. So it is not mandatory to care about this endpoint if you get everything about components to your monitoring system.

GET `/status/api/v1/cluster/{cluster_id}/service/{service_id}/`

Parameter	Type
`status`	integer

Possible values of the `status`:

Value	Meaning
0	All components in service have no problems
<any>	At least one component has a problem on some host

4.3 Heart Beat for Cluster

Status of cluster is an aggregation of service's status. There is no need to care about this if you already have information about services or components in your monitoring system.

GET `/status/api/v1/cluster/{cluster_id}/`

Parameter	Type
`status`	integer

Possible values of the `status`:

Value	Meaning
0	All services in cluster have no problems
<any>	At least one service has some problem

Глава 5

Ansible Modules and Plugins

We have a number of modules and plugins that allow developer to tweak ADCM internal states during playbook execution.

5.1 Object Multi State Set (adcm_multi_state_set)

- This is special ADCM only module which is useful for adding new multi_state for various ADCM objects.
- There is support of cluster, service, host, component and providers multi_states
- This one is allowed to be used in various execution contexts.
- It is allowed to change state of one service from another one, but it is dangerous. There is a risk of running parallel jobs that change the same state. Be careful.
- Using service_name give you ability to change one service from another or change service from component of another service.
- Using component_name give you ability to change component from another component of the same service or from component of another service on the same cluster (See examples 3 and 4)

5.1.1 Options

Required

- type *(string)* - type of object which should be changed. Take one of cluster, service, component, host or provider
- state *(string)* - value of state which should be set

Not required

- service_name *(string)* - useful in cluster,service and component context. In component context you can choose not matching service which is also belongs to the same cluster.
- component_name *(string)* - useful in cluster, service and component context. In that context you are able to set the multi_state for a component (it may be another component of another service belongs to that cluster)
- host_id - useful in provider context only. In that context you are able to set host multi_state from provider. See example 5

NOTE: It is allowed to change one service from another one, but it can be dangerous in case of parallel changing multi_state of one service.

5.1.2 Examples

```
- adcm_multi_state_set:
    type: "cluster"
    state: "bimba!"
```

```
- adcm_multi_state_set:
    type: "service"
    service_name: "First"
    state: "bimba!"
```

```
- adcm_multi_state_set:
    type: "component"
    component_name: "another_component"
    state: "bimba!"
```

```
- adcm_multi_state_set:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    state: "bimba!"
```

```
- adcm_multi_state_set:
    type: "host"
    host_id: {{host_id}}
    state: "bimba!"
```

5.1.3 Return values

```
state:
  returned: success
  type: str
  example: "operational"
```

5.2 Object Multi State Unset (adcm_multi_state_unset)

- This is special ADCM only module which is useful for changing multi_state for various ADCM objects.
- There is support of cluster, service, host, component and providers multi_states
- This one is allowed to be used in various execution contexts.
- It is allowed to change state of one service from another one, but it is dangerous. There is a risk of running parallel jobs that change the same state. Be careful.
- Using service_name give you ability to change one service from another or change service from component of another service.
- Using component_name give you ability to change component from another component of the same service or from component of another service (See examples 3 and 4)
- It raise error if there is no such multi_state on object

5.2.1 Options

Required

- type *(string)* - type of object which should be changed. Take one of cluster, service, component, host or provider
- state *(string)* - value of state which should be set

Not required

- service_name *(string)* - useful in cluster and component context. In component context you can choose not matching service which is also belongs to the same cluster.
- component_name *(string)* - useful in cluster, service and component context. In that context you are able to unset the multi_state for a component (it may be another component of another service belongs to that cluster)
- missing_ok *(boolean)* - if missing_ok is true then we shouldn't rise any exception if there is no such multi state on object. Default value is false.
- host_id - useful in provider context only. In that context you are able to unset host multi_state from provider. See example 5

NOTE: It is allowed to change one service from another one, but it can be dangerous in case of parallel changing multi_state of one service.

5.2.2 Examples

```
- adcm_multi_state_unset:
    type: "cluster"
    state: "bimba!"
```

```
- adcm_multi_state_unset:
    type: "service"
    service_name: "First"
    state: "bimba!"
```

```
- adcm_multi_state_unset:
    type: "component"
    component_name: "another_component"
    state: "bimba!"
```

```
- adcm_multi_state_unset:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    missing_ok: true
    state: "bimba!"
```

```
- adcm_multi_state_unset:
    type: "host"
    host_id: {{host_id}}
    state: "bimba!"
```

5.2.3 Return values

```
state:
  returned: success
  type: str
  example: "operational"
```

5.3 Object State Set (adcm_state)

- This is special ADCM only module which is useful for setting state for various ADCM objects.
- There is support of cluster, service, host and providers states
- This one is allowed to be used in various execution contexts.
- It is allowed to change state of one service from another one, but it is risky. There is a risk of running parallel jobs that change the same state. Be careful.
- Using service_name give you ability to change one service from another or change service from component of another service.
- Using component_name give you ability to change component from another component of the same service or from component of another service (See examples 3 and 4)

5.3.1 Options

Required

- type - Type of object which should be changed. Take one of cluster, service, component, provider, host
- state - Value of state which should be set.

Not Required

- service_name - Useful in cluster, service and component context. In component context you can choose not matching service which is belongs to the same cluster.
- component_name - Useful in cluster,service and component context. In that context you are able to set the state for a component (it may be another component of another service belongs to that cluster)
- host_id - useful in provider context only. In that context you are able to unset host state from provider. See example 5

NOTE: It is allowed to change one service from another one, but it can be dangerous in case of parallel changing state of one service.

5.3.2 Examples

```
- adcm_state:
    type: "cluster"
    state: "statey"
register: out
```

```
- adcm_state:
    type: "service"
    service_name: "First"
    state: "bimba!"
```

```
- adcm_state:
    type: "component"
    component_name: "another_component"
    state: "bimba!"
```

```
- adcm_state:
    type: "component"
    service_name: "another service"
```

```
    component_name: "another_component"
    missing_ok: true
    state: "bimba!"
```

```
- adcm_state:
    type: "host"
    host_id: {{host_id}}
    state: "bimba!"
```

5.3.3 Return values

```
state:
  returned: success
  type: str
  example: "operational"
```

5.4 Check Log Entry Add (adcm_check)

The adcm_check plugin is intended to log result of some checks to structured JSON-log storage. This log can be seen via **ADCM** user interface. Each invoke of adcm_check add one entry to JSON-log storage linked with **ADCM** job. You can invoke adcm_check with one job id any number of time per playbook.

5.4.1 Options

- `group_title` – Name of group check. Not required.
- `group_success_msg` – Description of success check or success results of check for group. Not required.
- `group_fail_msg` – Description of fail check or fail results of check for group. Not required.
- `title` – Name of check. Required.
- `result` – Result of check, yes/no. Required.
- `msg` – Description of check or results of check. Required, if no 'success_msg' and 'fail_msg' fields.
- `success_msg` – Description of success check or success results of check. Requires, if no 'msg' field.
- `fail_msg` – Description of fail check or fail results of check. Requires, if no 'msg' field.

5.4.2 Examples

```
- name: ADCM Check
  adcm_check:
    title: "Check"
    msg: "This is message"
    result: yes
```

```
- name: ADCM Check
  adcm_check:
    title: "Check"
    success_msg: "This is success message"
    fail_msg: "This is fail message"
    result: yes
```

```
- name: ADCM check
  adcm_check:
    group_title: "Group 1"
    group_success_msg: "This is success message"
    group_fail_msg: "This is fail message"
    title: "Check"
    msg: "This is message"
    result: yes
```

5.5 Custome Log Add (adcm_custom_log)

The adcm_custom_log plugin is intended to log result of some checks to structured JSON or TXT log storage. This log can be seen via **ADCM** user interface. Each invoke of adcm_custom_log add one entry to JSON or TXT log storage linked with **ADCM** job. You can invoke adcm_check with one job id any number of time per playbook.

5.5.1 Options

- `name` – Name of log. Required.
- `format` – Format of body, json/txt. Required.
- `path` – Path of file on node. Required, if field 'content' is none.
- `content` – Text. Required, if field 'path' is none.

5.5.2 Examples

```
- name: custom log
  adcm_custom_log:
    name: "Custom log json"
    format: "json"
    path: "/home/user/log.json"
```

```
- name: custom log
  adcm_custom_log:
    name: "Custom log txt"
    format: "txt"
    content: "This is log"
```

5.6 Config Set (adcm_config)

- This is special ADCM only module which is useful for setting of specified config key for various ADCM objects.
- There is support of cluster, service, component, host and providers config.
- This one is allowed to be used in various execution contexts.
- It is allowed to set config keys of one service from another one, but it is dangerous. There is a risk of running parallel jobs that change the same object. Be careful.
- Using service_name give you ability to change one service from another or change service from component of another service.
- Using component_name give you ability to change component from another component of the same service or from component of another service in the same cluster (See examples 3 and 4)

5.6.1 Options

Required

- type - Type of object which should be changed. Take one of cluster, service, component, provider, host
- key - Name of key which should be set
- value - value of key which should be set

Not Required

- service_name - Useful in cluster, service and component context. In component context you can choose not matching service which is belongs to the same cluster.
- component_name - Useful in cluster,service and component context. In that context you are able to set config for a component (it may be another component of another service belongs to that cluster)
- host_id - useful in provider context only. In that context you are able to set host config from provider. See example 5

NOTE: It is allowed to change one service from another one, but it can be dangerous in case of parallel changing config of one service.

5.6.2 Examples

```
- adcm_config:
    type: "service"
    service_name: "First"
    key: "some_int"
    value: 111
register: out
```

```
- adcm_config:
    type: "cluster"
    key: "some_map"
    value:
        key1: value1
        key2: value2
```

```
- adcm_state:
    type: "component"
    component_name: "another_component"
    key: "some_map"
    value: value
```

```
- adcm_state:
    type: "component"
    service_name: "another service"
    component_name: "another_component"
    key: "key"
    value: value
```

```
- adcm_state:
    type: "host"
    host_id: {{host_id}}
    key: "{{ config.key }}"
    value: "{{ config.value }}"
```

5.6.3 Return values

```
value:
  returned: success
  type: complex
```

5.7 Host Add (adcm_add_host)

- This module is intended to add new host to ADCM DB.
- This module should be run only in *host provider* context. Host Provider ID is taken from context.

5.7.1 Options

- `fqdn` – Fully Qualified Domain Name of new host.
- `description` – Optional description of new host.

5.7.2 Examples

```
- adcm_add_host:
    fqdn: my.host.org
    description: "my cool host"
```

5.7.3 Return values

```
result:
  host_id: 42
```

ID of new created host.

5.8 Host Delete (adcm_delete_host)

- This module is intended to delete host from ADCM DB.
- This module should be run only in **host** context. Host ID is taken from context.

5.8.1 Examples

```
- adcm_delete_host:
```

5.8.2 Return values

None

5.9 Host Add to Cluster (adcm_remove_host_from_cluster)

- This module is intended to remove host from cluster in ADCM DB.
- This module should be run only in *cluster* or *service* context. Cluster ID is taken from context.

5.9.1 Options

You should specify `fqdn` or `host_id` of removed host:

- `fqdn` – Fully Qualified Domain Name of removed host;
- `host_id` – Host ID of removed host in ADCM DB.

5.9.2 Examples

```
- adcm_remove_host_from_cluster:
    fqdn: my.host.org
```

```
- adcm_remove_host_from_cluster:
    host_id: 42
```

5.9.3 Return values

None

5.10 Host Remove from Cluster (adcm_add_host_to_cluster)

- This module is intended to add existing host to cluster in ADCM DB.
- This module should be run only in **cluster** or **service** context. Cluster ID is taken from context.

5.10.1 Options

You should specify fqdn or host_id of added host

- fqdn - Fully Qualified Domain Name of new added host
- host_id - Host ID of added host in ADCM DB.

5.10.2 Examples

```
- adcm_add_host_to_cluster:
    fqdn: my.host.org
```

```
- adcm_add_host_to_cluster:
    host_id: 42
```

5.10.3 Return values

None

5.11 Host Component Map Set (adcm_hc)

- This module is intended to change host component map (hc) for cluster. In the sequence of operations you can break component constraints. All constraints are checked only once, when all operations are performed.
- This module should be run only in *cluster* or *service* context. Cluster ID is taken from context.

5.11.1 Examples

```
- name: move master component from host1.company.com to host2.company.com
  adcm_hc:
    operations:
      -
        action: "remove"
        service: "hadoop"
        component: "master"
        host: "host.company.com"
      -
        action: "add"
        service: "hadoop"
        component: "master"
        host: "host2.company.com"
```

5.11.2 Return values

None.

5.12 Service Delete (adcm_delete_service)

- This module is intended to delete service from ADCM DB.
- This module should be run only in **service** context. Service ID is taken from context.

5.12.1 Examples

```
- adcm_delete_service:
```

5.12.2 Return values

None

Глава 6

Adcm bundles packer

Adcm_sdk_pack – part of adcm_client package. Purposes that are resolved:

- Build .tgz files from dirs with sources;
- Set version of bundle and name of built artifact in ci circle;
- Support spec files.

```
usage: adcm_sdk_pack [-h] -n NAME -p PATH [-t TARPATH] [-w WORKSPACE] [-c]
           [-m MASTER_BRANCHES [MASTER_BRANCHES ...]]

Builds bundles from sources.

optional arguments:
  -h, --help            show this help message and exit
  -n NAME, --name NAME  bundle name
  -p PATH, --path PATH  path where to seek configs
  -t TARPATH, --tarpath TARPATH
            path where to save
  -w WORKSPACE, --workspace WORKSPACE
            workspace where to build
  -c, --dont-clean-ws   remove temporary files from ws
  -m MASTER_BRANCHES [MASTER_BRANCHES ...], --master-branches MASTER_BRANCHES [MASTER_BRANCHES ...]
            non default master branches
```

6.1 Naming rules

6.2 Spec files

```
---
version: "1.0"
editions:
 - name: None
   exclude:
   - tests/
   - build/
   - _config.yaml
   - pylintrc
   - (.*/)?(\..+)
   - .*README\.md
```

```
  - spec.yaml
  - tests_token
  - module_struct.py
  - requirements.yml
  - yandexcloud_module/
  - tests\.sh

  preprocessors:
  - type: script
    script: build/config_builder.py
    args:
    - _config.yaml

  - type: python_mod_req
    requirements: requirements.yml
    prepared_image: ci.arenadata.io/adcm:yandexcloud  # for local debug purposes
    image: arenadata/adcm:2020013015

  - type: script
    script: ./module_struct.py

  - type: splitter
    files:
     - EULA.txt.j2
```

Version 1.0 functionality:

- Exclude files and dir from being packed by regular expression.
- Process prepack actions. Types:
  - `script` – execute script from repository. Params:
    - script: path to script;
    - args: list of arguments that are passed to script.
  - `python_mod_req` – install python-package to bundle. Params:
    - requirements: file with required packages. Format:

```
# List of utils that are not present in image to proceed
# packages build and must be installed before python packages.
# Not required.
system_pkg:
  - gcc
  - g++
  - linux-headers
# List of python packages.
# Required.
python_mod:
  - yandexcloud==0.20.0
```

    - image: image that will be used as base to install packages;
    - prepared_image: for local debug purposes. An image prepared from image specified in image parameter by installation of required python-packages. Used to optimize build time in case of local tests running.
  - `splitter` – generate file from jinja2 templates. Variable "edition" is set to current edition name parameter. New file name is obtained by removing ".j2" file extension from template name.