

---

# Armada Documentation

**Armada Team**

**Dec 17, 2025**



# ARMADA

<b>1 Armada</b>	<b>1</b>
1.1 Overview . . . . .	1
1.2 Components . . . . .	1
1.3 Installation . . . . .	1
1.4 Integration Points . . . . .	2
1.5 Further Reading . . . . .	2
<b>Index</b>	<b>65</b>



## ARMADA

Armada is a tool for managing multiple Helm charts with dependencies by centralizing all configurations in a single Armada YAML and providing life-cycle hooks for all Helm releases.

Find more documentation for Armada in the [Armada documentation](#).

### 1.1 Overview

The Armada Python library and command line tool provide a way to synchronize a Helm target with an operator's intended state, consisting of several charts, dependencies, and overrides using a single file or directory with a collection of files. This allows operators to define many charts, potentially with different namespaces for those releases, and their overrides in a central place. With a single command, deploy and/or upgrade them where applicable.

Armada also supports fetching Helm chart source and then building charts from source from various local and remote locations, such as Git endpoints, tarballs or local directories.

It will also give the operator some indication of what is about to change by assisting with diffs for both values, values overrides, and actual template changes.

Its functionality extends beyond Helm, assisting in interacting with Kubernetes directly to perform basic pre- and post-steps, such as removing completed or failed jobs, running backup jobs, blocking on chart readiness, or deleting resources that do not support upgrades. However, primarily, it is an interface to support orchestrating Helm.

### 1.2 Components

Armada consists of two separate but complementary components:

1. CLI component (**mandatory**) which interfaces directly with [Helm](#).
2. API component (**optional**) which services user requests through a wsgi server (which in turn communicates with the [Helm CLI](#)) and provides the following additional functionality:
  - Role-Based Access Control.
  - Limiting projects to specific functionality by leveraging project-scoping provided by [Keystone](#).

### 1.3 Installation

#### 1.3.1 Quick Start (via Container)

Armada can be most easily installed as a container, which requires Docker to be executed. To install Docker, please reference the following [install guide](#).

Afterward, you can launch the Armada container by executing:

```
$ sudo docker run -d --net host -p 8000:8000 --name armada \  
-v ~/.kube/config:/armada/.kube/config \  
-v $(pwd)/examples/./examples quay.io/airshipit/armada:latest-ubuntu_bionic
```

### 1.3.2 Manual Installation

For a comprehensive manual installation guide, please see [Manual Install Guide](#).

### 1.3.3 Usage

To run Armada, simply supply it with your YAML-based intention for any number of charts:

```
$ armada apply examples/openstack-helm.yaml [ --debug ]
```

Which should output something like this:

```
$ armada apply examples/openstack-helm.yaml 2017-02-10 09:42:36,753  
  
armada INFO Cloning git:  
...
```

For more information on how to install and use Armada, please reference: [Armada Quickstart](#).

## 1.4 Integration Points

Armada CLI component has the following integration points:

- [Helm](#) manages Armada chart installations.
- [Deckhand](#) is one of the supported control document sources for Armada.
- [Prometheus](#) exporter is provided for metric data related to application of charts and collections of charts. See [metrics](#).

In addition, Armada's API component has the following integration points:

- [Keystone](#) (OpenStack's identity service) provides authentication and support for role-based authorization.

## 1.5 Further Reading

[Airship](#).

### 1.5.1 Developers Guide

#### Developer Install Guide

#### Quick Start (via Container)

#### Note

If actively developing new Armada functionality, it is recommended to proceed with *Manual Installation* instead.

To use the docker container to develop:

1. Clone the [Armada repository](#).
2. `cd` into the cloned directory.

```
$ git clone https://opendev.org/airship/armada.git && cd armada
```

3. Next, run the following commands to install `tox`, generate sample policy and configuration files, and build Armada charts as well as the Armada container image. Armada `Dockerfile.DISTRO` files are located in `images/armada`. Supported DISTROs are `ubuntu_bionic` and `opensuse_leap15`. By default, DISTRO is `ubuntu_bionic`.

```
$ pip install tox
$ tox -e genconfig
$ tox -e genpolicy
$ export DISTRO=distro_name
$ docker build -f Dockerfile.${DISTRO} -t armada/latest
$ make images
```

4. Run the container via Docker:

```
$ docker run -d --name armada -v ~/.kube/:/armada/.kube/ -v $(pwd)/etc:/etc_
↪armada:local
```

#### Note

The first build will take several minutes. Afterward, it will build much faster.

## Manual Installation

### Pre-requisites

The guide below consolidates the installation of all pre-requisites. For help troubleshooting individual resources, reference their installation guides.

Armada requires a Kubernetes cluster to be deployed, along with [kubectl](#) and [Helm](#).

1. Install Kubernetes (k8s) and deploy a k8s cluster.  
Reference the *Kubernetes Cluster Management* section below for help.
2. Install and configure [kubectl](#)
3. Ensure that `~/.kube/config` exists and is properly configured by executing:

```
$ kubectl config view
```

If the file does not exist, please create it by running:

```
$ kubectl
```

4. Install and configure the [Helm](#) client.

### Kubernetes Cluster Management

To test Armada fixes/features a Kubernetes cluster must be installed.

Either software is recommended:

- [Kubeadm](#)
- [Kubeadm-AIO](#)

### Armada CLI Installation

Follow the steps below to install the Armada CLI.

#### Note

Some commands below use `apt-get` as the package management software. Use whichever command corresponds to the Linux distro being used.

#### Warning

Armada is tested against a Ubuntu 16.04 and Opensuse(leap15.1)environment.

Clone the Armada repository, cd into it:

```
git clone https://opendev.org/airship/armada.git && cd armada
```

It is recommended that Armada be run inside a virtual environment. To do so:

```
$ virtualenv -p python3 venv
...
>> New python executable in <...>/venv/bin/python3
```

Afterward, source the executable:

```
source <...>/venv/bin/activate
```

Next, ensure that `pip` is installed.

```
$ apt-get install -y python3-pip $ pip3 install --upgrade pip
```

Finally, run (from inside the Armada root directory):

```
$ (venv) make build
```

The above command will install `pip` requirements and execute `python setup.py build` within the virtual environment.

Verify that the Armada CLI is installed:

```
$ armada --help
```

Which should emit:

```
>> Usage: armada [OPTIONS] COMMAND [ARGS]...
>>
>> Multi Helm Chart Deployment Manager
...

```

## Armada API Server Installation

The Armada API server is not required in order to use the Armada CLI, which in this sense is standalone. The Armada CLI communicates with the Helm CLI. The Armada API server and CLI interface have the exact same functionality. However, the Armada API server offers the following additional functionality:

- Role-Based Access Control, allowing Armada to provide authorization around specific Armada functionality.
- [Keystone](#) authentication and project scoping, providing an additional layer of security.

Before proceeding, ensure that the steps in [Armada CLI Installation](#) have been followed.

1. Determine where the Armada configuration/deployment files should be stored. The default location is `/etc/armada`. To override the default, run:

```
$ export OS_ARMADA_CONFIG_DIR=<desired_path>
```

2. If the directory specified by `OS_ARMADA_CONFIG_DIR` is empty, run (from the Armada root directory):

```
$ cp etc/armada/* <OS_ARMADA_CONFIG_DIR>/
$ mv <OS_ARMADA_CONFIG_DIR>/armada.conf.sample <OS_ARMADA_CONFIG_DIR>/armada.conf
```

# Install uwsgi:

```
$ apt-get install uwsgi -y
```

1. Ensure that port 8000 is available or else change the `PORT` value in `entrypoint.sh`.
2. From the root Armada directory, execute:

```
$ ./entrypoint.sh server
```

3. Verify that the Armada server is running by executing:

```
$ TOKEN=$(openstack token issue --format value -c id)
$ curl -i -X GET localhost:8000/versions -H "X-Auth-Token: $TOKEN"
```

Note that the port above uses the default value in `entrypoint.sh`.

## Development Utilities

Armada comes equipped with many utilities useful for developers, such as unit test or linting jobs.

Many of these commands require that `tox` be installed. To do so, run:

```
$ pip3 install tox
```

To run the Python linter, execute:

```
$ tox -e pep8
```

(continues on next page)

(continued from previous page)

```
or
```

```
$ make test-pep8
```

To lint Helm charts, execute:

```
$ make lint
```

To run unit tests, execute:

```
$ tox -e py35
```

```
or
```

```
$ make test-unit
```

To run the test coverage job:

```
$ tox -e coverage
```

```
or
```

```
$ make test-coverage
```

To run security checks via [Bandit](#) execute:

```
$ tox -e bandit
```

```
or
```

```
$ make test-bandit
```

To build the docker images:

```
$ make images
```

To build all Armada charts, execute:

```
$ make charts
```

To build a helm template for the charts:

```
$ make dry-run
```

To run lint, charts, and image targets all at once:

```
$ make all
```

To render any documentation that has build steps:

```
$ tox -e docs
```

```
or
```

(continues on next page)

(continued from previous page)

```
$ make docs
```

To build armada's image:

```
$ make run_armada
```

To build all images:

```
$ make run_images
```

To generate sample configuration and policy files needed for Armada deployment, execute (respectively):

```
$ tox -e genconfig  
$ tox -e genpolicy
```

## Troubleshooting

The error messages are included in bullets below and tips to resolution are included beneath each bullet.

- “FileNotFoundError: [Errno 2] No such file or directory: ‘etc/armada/api-paste.ini’”

Reason: this means that Armada is trying to instantiate the server but failing to do so because it can't find an essential configuration file.

Solution:

```
$ cp etc/armada/armada.conf.sample /etc/armada/armada.conf
```

This copies the sample Armada configuration file to the appropriate directory.

- For any errors related to `tox`:

Ensure that `tox` is installed:

```
$ sudo apt-get install tox -y
```

- For any errors related to running `tox -e py35`:

Ensure that `python3-dev` is installed:

```
$ sudo apt-get install python3-dev -y
```

## Contribution Guidelines

If you would like to contribute to the development of OpenStack, you must follow the steps in this page:

<https://docs.openstack.org/infra/manual/developers.html>

If you already have a good understanding of how the system works and your OpenStack accounts are set up, you can skip to the development workflow section of this documentation to learn how changes to OpenStack should be submitted for review via the Gerrit tool:

<https://docs.openstack.org/infra/manual/developers.html#development-workflow>

Pull requests submitted through GitHub will be ignored.

## 1.5.2 Operations Guide

### Document Authoring Guide

v1

#### v1 Authoring

##### armada/Manifest/v1


keyword	type	action
release_pref:	string	appends to the front of all charts released by the manifest in order to manage releases throughout their lifecycle
chart_groups	array	references ChartGroup document of all groups

#### Manifest Example

```
---
schema: armada/Manifest/v1
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - chart_group
```

##### armada/ChartGroup/v1

keyword	type	action
description	string	description of chart set
chart_group	array	reference to chart document
sequenced	bool	enables sequenced chart deployment in a group
test_charts	bool	run pre-defined helm tests in a ChartGroup (DEPRECATED)

 **Danger**

DEPRECATION: The test\_charts key will be removed, as Armada will run helm tests for all charts by default.

#### Chart Group Example

```
---
schema: armada/ChartGroup/v1
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
```

(continues on next page)

(continued from previous page)

```

description: Deploys Simple Service
sequenced: False
chart_group:
  - chart
  - chart

```

## armada/Chart/v1

### Danger

DEPRECATION: `timeout` key-value will be removed `timeout` will be defined under `wait` object.

## Chart

key-word	type	action
<code>chart_name</code>	string	name for the chart
<code>release</code>	string	name of the release (Armada will prepend with <code>release-prefix</code> during processing)
<code>namespace</code>	string	namespace of your chart
<code>wait</code>	object	See <i>Wait</i> .
<code>protected</code>	object	do not delete FAILED releases when encountered from previous run (provide the 'continue_processing' bool to continue or halt execution (default: halt))
<code>test</code>	object	See <i>Test</i> .
<code>install</code>	object	install the chart into your Kubernetes cluster
<code>upgrade</code>	object	upgrade the chart managed by the armada yaml
<code>delete</code>	object	See <i>Delete</i> .
<code>values</code>	object	override any default values in the charts
<code>source</code>	object	provide a path to a <code>git repo</code> , <code>local dir</code> , or <code>tarball url</code> chart
<code>dependencies</code>	object	(optional) Override the <code>builtin chart dependencies</code> with a list of Chart documents to use as dependencies instead. NOTE: Builtin ".tgz" dependencies are not yet supported.
<code>timeout</code>	int	time (in seconds) allotted for chart to deploy when 'wait' flag is set (DEPRECATED)

## Wait

key-word	type	action
time-out	int	time (in seconds) to wait for chart to deploy
resources	array	Array of <i>Wait Resource</i> to wait on, with labels added to each item. Defaults to pods and jobs (if any exist) matching labels.
labels	object	Base mapping of labels to wait on. They are added to any labels in each item in the resources array.
native	boolean	See <i>Wait Native</i> .

## Wait Resource

key-word	type	action
type	string	k8s resource type, supports: controllers ('deployment', 'daemonset', 'statefulset'), 'pod', 'job'
labels	object	mapping of kubernetes resource labels
min_ready	int string	Only for controller type`s. Amount of pods in a controller which must be ready. Can be integer or percent string e.g. ``80%. Default 100%.

## Wait Native

Config for the native helm (install|upgrade) --wait flag.

keyword	type	action
enabled	boolean	defaults to true

## Test

Run helm tests on the chart after install/upgrade.

keyword	type	action
enabled	bool	whether to enable/disable helm tests for this chart (default True)
timeout	int	time (in sec) to wait for completion of Helm tests. Default 300.
options	object	See <i>Test Options</i> .

### Note

Armada will attempt to run helm tests by default. They may be disabled by setting the enabled key to False.

### Danger

**DEPRECATION:** In addition to an object with the above fields, the `test` key currently also supports `bool`, which maps to `enabled`, but this is deprecated and will be removed. The `cleanup` option below is set to `true` in this case for backward compatibility.

## Test Options

Test options to pass through directly to helm.

keyword	type	action
<code>cleanup</code>	<code>bool</code>	cleanup test pods after test completion, defaults to false

### Note

If `cleanup` is `true` this prevents being able to debug a test in the event of failure.

Historically, the preferred way to achieve test cleanup has been to add a pre-upgrade delete action on the test pod.

This still works, however it is usually no longer necessary as Armada now automatically cleans up any test pods which match the `wait.labels` of the chart, immediately before running tests. Similar suggestions have been made for how `helm test --cleanup` itself ought to work (<https://github.com/helm/helm/issues/3279>).

## Upgrade - Pre

keyword	type	action
<code>pre</code>	<code>object</code>	actions performed prior to updating a release

## Upgrade - Actions

keyword	type	action
<code>update</code>	<code>object</code>	update daemonsets in pre-upgrade update actions
<code>delete</code>	<code>sequence</code>	delete jobs and pods in pre-upgrade delete actions

## Upgrade - Actions - Update/Delete

keyword	type	action
<code>name</code>	<code>string</code>	name of action
<code>type</code>	<code>string</code>	type of Kubernetes workload to execute in scope for action
<code>labels</code>	<code>object</code>	k:v mapping of labels to select Kubernetes resources

### Note

Update Actions only support type: `'daemonset'`

**Note**

Delete Actions support type: 'pod', 'job', 'cronjob'

**Chart Example**

```

---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1
  release: blog-1
  namespace: default
  wait:
    timeout: 100
  protected:
    continue_processing: false
  test:
    enabled: true
  install:
    no_hooks: false
  upgrade:
    no_hooks: false
  pre:
    update:
      - name: test-daemonset
        type: daemonset
        labels:
          foo: bar
          component: bar
          rak1: enabled
    delete:
      - name: test-job
        type: job
        labels:
          foo: bar
          component: bar
          rak1: enabled
  values: {}
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: .
    reference: master

```

**Delete**

keyword	type	action
timeout	integer	time (in seconds) to wait for chart to be deleted

## Source

keyword	type	action
type	string	source to build the chart: <code>git</code> , <code>local</code> , or <code>tar</code>
location	string	url or path to the chart's parent directory
subpath	string	(optional) relative path to target chart from parent ( <code>.</code> if not specified)
reference	string	(optional) branch, commit, or reference in the repo ( <code>master</code> if not specified)
proxy_server	string	(optional) proxy server URL for downloading <code>git</code> or <code>tar</code> charts

## Source Example

```
# type git
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1
  release: blog-1
  namespace: default
  wait:
    timeout: 100
    labels:
      component: blog
  install:
    no_hooks: false
  upgrade:
    no_hooks: false
  values: {}
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: .
    reference: master

# type local
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1
  release: blog-1
  namespace: default
  wait:
    timeout: 100
  install:
    no_hooks: false
  upgrade:
```

(continues on next page)

```
no_hooks: false
values: {}
source:
  type: local
  location: /path/to/charts
  subpath: chart
  reference: master

# type tar
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1
  release: blog-1
  namespace: default
  wait:
    timeout: 100
  install:
    no_hooks: false
  upgrade:
    no_hooks: false
  values: {}
  source:
    type: tar
    location: https://localhost:8879/charts/chart-0.1.0.tgz
    subpath: mariadb
    reference: null
    proxy_server: http://my.proxy.server:8888
```

## Defining a Manifest

To define your Manifest you need to define a `armada/Manifest/v1` document, `armada/ChartGroup/v1` document, `armada/Chart/v1`. Following the definitions above for each document you will be able to construct an armada manifest.

## Armada - Deploy Behavior

1. Armada will perform set of pre-flight checks to before applying the manifest - validate input manifest - check chart source locations are valid
2. Deploying Armada Manifest
  1. If the chart is not found
    - we will install the chart
  3. If exist then
    - Armada will check if there are any differences in the chart
    - if the charts are different then it will execute an upgrade
    - else it will not perform any actions

**Note**

You can use references in order to build your charts, this will reduce the size of the chart definition will show example in multichart below

**Simple Example**

```

---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1
  release: blog-1
  namespace: default
  values: {}
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: blog-1
    reference: new-feat
---
schema: armada/ChartGroup/v1
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
  description: Deploys Simple Service
  sequenced: False
  chart_group:
    - blog-1
---
schema: armada/Manifest/v1
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - blog-group

```

**Multichart Example**

```

---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1

```

(continues on next page)

```
release: blog-1
namespace: default
values: {}
source:
  type: git
  location: https://github.com/namespace/repo
  subpath: blog1
  reference: master
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-2
data:
  chart_name: blog-2
  release: blog-2
  namespace: default
  values: {}
  source:
    type: tar
    location: https://github.com/namespace/repo/blog2.tgz
    subpath: blog2
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-3
data:
  chart_name: blog-3
  release: blog-3
  namespace: default
  values: {}
  source:
    type: local
    location: /home/user/namespace/repo/blog3
---
schema: armada/ChartGroup/v1
metadata:
  schema: metadata/Document/v1
  name: blog-group-1
data:
  description: Deploys Simple Service
  sequenced: False
  chart_group:
    - blog-2
---
schema: armada/ChartGroup/v1
metadata:
  schema: metadata/Document/v1
  name: blog-group-2
data:
  description: Deploys Simple Service
```

(continues on next page)

(continued from previous page)

```

sequenced: False
chart_group:
  - blog-1
  - blog-3
---
schema: armada/Manifest/v1
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - blog-group-1
    - blog-group-2

```

### Dependency Override Example

```

---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  chart_name: blog-1
  release: blog-1
  namespace: default
  values: {}
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: blog-1
    reference: new-feat
  dependencies:
    - blog-1-dep
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1-dep
data:
  chart_name: blog-1-dep
  release: blog-1-dep
  namespace: default
  values: {}
  source:
    type: git
    location: https://github.com/namespace/dep-repo
    subpath: blog-1-dep
    reference: new-feat
---
schema: armada/ChartGroup/v1

```

(continues on next page)

(continued from previous page)

```
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
  description: Deploys Simple Service
  sequenced: False
  chart_group:
    - blog-1
---
schema: armada/Manifest/v1
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - blog-group
```

## References

For working examples please check the examples in our repo [here](#).

## v1 Schemas

Below are the schemas Armada uses to validate Charts, Chart Groups, and Manifests.

### Charts

Charts consist of the smallest building blocks in Armada. A Chart is comparable to a Helm chart. Charts consist of all the labels, dependencies, install and upgrade information, hooks and additional information needed to convey to Helm.

### Chart Groups

A Chart Group consists of a list of charts. Chart Group documents are useful for managing a group of Chart documents together.

### Manifests

A Manifest is the largest building block in Armada. Manifest documents are responsible for managing collections of Chart Group documents.

## Validation Schemas

### Introduction

All schemas below are [Deckhand DataSchema](#) documents, which are essentially JSON schemas, with additional meta-data useful for Deckhand to perform [layering](#) and [substitution](#).

The validation schemas below are used by Armada to validate all ingested Charts, Chart Groups, and Manifests. Use the schemas below as models for authoring Armada documents.

## Schemas

- Chart schema.

JSON schema against which all documents with `armada/Chart/v1 metadata.name` are validated.

Listing 1: Schema for `armada/Chart/v1` documents.

```
# NOTE: Do not modify this schema, it is deprecated.
---
schema: deckhand/DataSchema/v1
metadata:
  name: armada/Chart/v1
  schema: metadata/Control/v1
data:
  $schema: http://json-schema.org/schema#
  definitions:
    labels:
      type: object
      additionalProperties:
        type: string
    hook_action:
      type: array
    items:
      properties:
        name:
          type: string
        type:
          type: string
        labels:
          $ref: '#/definitions/labels'
      required:
        - type
      additionalProperties: false
  type: object
  properties:
    release:
      type: string
    chart_name:
      type: string
    namespace:
      type: string
    values:
      type: object
    dependencies:
      type: array
    items:
      type: string
    protected:
      type: object
      properties:
        continue_processing:
          type: boolean
      additionalProperties: false
```

(continues on next page)

```
test:
  anyOf:
    - type: boolean
    - type: object
    properties:
      enabled:
        type: boolean
      timeout:
        type: integer
      options:
        type: object
        properties:
          cleanup:
            type: boolean
          additionalProperties: false
      additionalProperties: false
  timeout:
    type: integer
  wait:
    type: object
    properties:
      timeout:
        type: integer
    resources:
      type: array
      items:
        properties:
          type:
            type: string
          labels:
            $ref: '#/definitions/labels'
          min_ready:
            anyOf:
              - type: integer
              - type: string
          required:
            - type
          additionalProperties: false
        labels:
          $ref: "#/definitions/labels"
        # Config for helm's native `--wait` param.
      native:
        type: object
        properties:
          # TODO: Add separate timeout for native wait?
          enabled:
            type: boolean
          additionalProperties: false
    additionalProperties: false
  source:
    type: object
    properties:
```

(continues on next page)

(continued from previous page)

```
    type:
      type: string
    location:
      type: string
    subpath:
      type: string
    reference:
      type: string
    proxy_server:
      type: string
    auth_method:
      type: string
    required:
      - location
      - subpath
      - type
  delete:
    type: object
    properties:
      timeout:
        type: integer
  install:
    # NOTE(sh8121att) Not clear that this key is actually used
    # in the code. Will leave it here for backward compatibilities
    # until an additional audit is done.
    type: object
  upgrade:
    type: object
    properties:
      no_hooks:
        type: boolean
    pre:
      type: object
      additionalProperties: false
      properties:
        delete:
          $ref: '#/definitions/hook_action'
        update:
          $ref: '#/definitions/hook_action'
        create:
          $ref: '#/definitions/hook_action'
    post:
      type: object
      additionalProperties: false
      properties:
        create:
          $ref: '#/definitions/hook_action'
    options:
      type: object
      properties:
        force:
          type: boolean
```

(continues on next page)

(continued from previous page)

```

    additionalProperties: false
  required:
    - no_hooks
  additionalProperties: false
required:
  - namespace
  - chart_name
  - release
  - source
additionalProperties: false
...

```

This schema is used to sanity-check all Chart documents that are passed to Armada.

- Chart Group schema.

JSON schema against which all documents with `armada/Chart/v1` `metadata.name` are validated.

Listing 2: Schema for `armada/ChartGroup/v1` documents.

```

# NOTE: Do not modify this schema, it is deprecated.
---
schema: deckhand/DataSchema/v1
metadata:
  name: armada/ChartGroup/v1
  schema: metadata/Control/v1
data:
  $schema: http://json-schema.org/schema#
  properties:
    name:
      type: string
    description:
      type: string
    sequenced:
      type: boolean
    # TODO(MarshM): Deprecate `test_charts`, it is no longer useful
    test_charts:
      type: boolean
    chart_group:
      type: array
      items:
        type: string
  required:
    - chart_group
  additionalProperties: false
...

```

This schema is used to sanity-check all Chart Group documents that are passed to Armada.

- Manifest schema.

JSON schema against which all documents with `armada/Manifest/v1` `metadata.name` are validated.

Listing 3: Schema for armada/Manifest/v1 documents.

```
# NOTE: Do not modify this schema, it is deprecated.
---
schema: deckhand/DataSchema/v1
metadata:
  name: armada/Manifest/v1
  schema: metadata/Control/v1
data:
  $schema: http://json-schema.org/schema#
  properties:
    release_prefix:
      type: string
    chart_groups:
      type: array
      items:
        type: string
  required:
    - chart_groups
    - release_prefix
  additionalProperties: false
...
```

This schema is used to sanity-check all Manifest documents that are passed to Armada.

## Authoring Guidelines

All Armada documents must use the deckhand/DataSchema/v1 schema.

## v2 (EXPERIMENTAL!)

### v2 Authoring

#### Danger

EXPERIMENTAL: v2 docs are still experimental and WILL have breaking changes before they are finalized.

### armada/Manifest/v2

keyword	type	action
release_pref:	string	appends to the front of all charts released by the manifest in order to manage releases throughout their lifecycle
chart_groups	array	A list of the metadata.name of each ChartGroup to deploy in order.

## Manifest Example

```
---
schema: armada/Manifest/v2
```

(continues on next page)

(continued from previous page)

```
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - chart_group
```

### armada/ChartGroup/v2

keyword	type	action
description	string	description of chart set
chart_group	array	A list of the metadata.name of each Chart to deploy.
sequenced	bool	If true, deploys each chart in sequence, else in parallel. Default false.

### Chart Group Example

```
---
schema: armada/ChartGroup/v2
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
  description: Deploys Simple Service
  chart_group:
    - chart1
    - chart2
```

### armada/Chart/v2

## Chart

key-word	type	action
release	string	name of the release (Armada will prepend with <code>release-prefix</code> during processing)
names- pace	string	namespace of your chart
wait	ob- ject	See <i>Wait</i> .
pro- tected	ob- ject	do not delete FAILED releases when encountered from previous run (provide the 'con- tinue_processing' bool to continue or halt execution (default: halt))
test	ob- ject	See <i>Test</i> .
up- grade	ob- ject	See <i>Upgrade</i> .
delete	ob- ject	See <i>Delete</i> .
values	ob- ject	(optional) override any default values in the charts
source	ob- ject	provide a path to a <code>git repo</code> , <code>local dir</code> , or <code>tarball url</code> chart
depen- dencies	ob- ject	(optional) Override the <a href="#">builtin chart dependencies</a> with a list of Chart documents to use as depen- dencies instead. NOTE: Builtin “.tgz” dependencies are not yet supported.

## Wait

keyword	type	action
timeout	int	time (in seconds) to wait for chart to deploy
resources	dict   array	<p><i>Wait Resource</i> s to wait on. Defaults to all supported resource types (see <i>Wait Resource .type</i>), with <code>required: false</code>.</p> <p><b>dict</b> - Maps resource types to one of:</p> <ul style="list-style-type: none"> <li><i>Wait Resource</i> without <code>.type</code> (single config)</li> <li>list of <i>Wait Resource</i> without <code>.type</code> (multiple configs)</li> <li><code>false</code> (disabled)</li> </ul> <p>Any resource type not overridden retains its default config mentioned above.</p> <p><b>array</b> - Lists all <i>Wait Resource</i> s to use, completely overriding the default. Can be set to [] to disable all resource types.</p> <p>See also <i>Wait Resources Examples</i>.</p>
labels	object	Base mapping of labels to wait on. They are added to any labels in each item in the <code>resources</code> array.
native	boolean	See <i>Wait Native</i> .

## Wait Resource

key-word	type	action
type	string	K8s resource type, supports: 'deployment', 'daemonset', 'statefulset', 'pod', 'job'. NOTE: Omit when <i>Wait .resources</i> is a dict, as then the dict key is used instead.
labels	object	Kubernetes labels specific to this resource. <i>Wait .labels</i> are included with these, so only define this if additional labels are needed to identify the targeted resources.
min_read	int   string	Only for controller <i>type</i> s. Amount of pods in a controller which must be ready. Can be integer or percent string e.g. <code>80%</code> . Default <code>100%</code> .
al-low_asyn	boolean	Only for <code>daemonset</code> and <code>statefulset</code> types. Whether to wait for async update strategies, i.e. <code>OnDelete</code> or <code>partitioned RollingUpdate</code> . Defaults to <code>false</code> in order to fail fast in cases where the async update is not expected to complete until same point later on.
re-quired	boolean	Whether to require the resource to be found. Defaults to <code>true</code> for explicit items in <code>`wait.resources</code> . See <code>wait.resources</code> for its overall defaults.

## Wait Resources Examples

```
wait:
# ...
# Disable all waiting.
resources: []
```

```
wait:
# ...
# Disable waiting for a given type (job).
resources:
  job: false
```

```
wait:
# ...
# Use min_ready < 100%.
resources:
  daemonset:
    min_ready: 80%
```

```
wait:
resources:
# Multiple configs for same type.
  daemonset:
    - labels:
      component: one
      min_ready: 80%
    - labels:
      component: two
      min_ready: 50%
```

```
wait:
# ...
resources:
  - type: daemonset
    labels:
      component: critical
    min_ready: 100%
  - type: daemonset
    labels:
      component: best_effort
    min_ready: 80%
# ... (re-include any other resource types needed when using list)
```

## Wait Native

Config for the native helm (install|upgrade) --wait flag.

keyword	type	action
enabled	boolean	defaults to false

## Test

Run helm tests on the chart after install/upgrade.

keyword	type	action
enabled	bool	whether to enable/disable helm tests for this chart (default True)
timeout	int	time (in sec) to wait for completion of Helm tests. Default 300.
options	object	See <i>Test Options</i> .

### **Note**

Armada will attempt to run helm tests by default. They may be disabled by setting the `enabled` key to `false`.

## Test Options

Test options to pass through directly to helm.

keyword	type	action
cleanup	bool	Same as Helm CLI.

### **Note**

If `cleanup` is `true` this prevents being able to debug a test in the event of failure.

Historically, the preferred way to achieve test cleanup has been to add a pre-upgrade delete action on the test pod.

This still works, however it is usually no longer necessary as Armada now automatically cleans up any test pods which match the `wait.labels` of the chart, immediately before running tests. Similar suggestions have been made for how `helm test --cleanup` itself ought to work (<https://github.com/helm/helm/issues/3279>).

## Upgrade

keyword	type	action
options	object	See <i>Upgrade - Options</i> .
pre	object	See <i>Upgrade - Pre</i> .

## Upgrade - Options

Upgrade options to pass through directly to helm.

keyword	type	action
no_hooks	boolean	Same as Helm CLI.
force	boolean	Same as Helm CLI.

## Upgrade - Pre

keyword	type	action
delete	array	List of <i>Upgrade - Pre - Delete</i> .

## Upgrade - Pre - Delete

keyword	type	action
type	string	type of kubernetes resource to delete supported types are: 'pod', 'job', 'cronjob'.
labels	object	k:v mapping of labels to select Kubernetes resources

## Chart Example

```

---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  release: blog-1
  namespace: default
  wait:
    timeout: 100
  protected:
    continue_processing: false
  test:
    enabled: true
  upgrade:
    pre:
      delete:
        - name: test-job
          type: job
          labels:
            foo: bar
            component: bar
            rak1: enabled
  source:
    type: git
    location: https://github.com/namespace/repo
    reference: master

```

## Delete

keyword	type	action
timeout	integer	time (in seconds) to wait for chart to be deleted

## Source

keyword	type	action
type	string	source to build the chart: <code>git</code> , <code>local</code> , or <code>tar</code>
location	string	url or path to the chart's parent directory
subpath	string	(optional) relative path to target chart from parent ( <code>.</code> if not specified)
reference	string	(optional) branch, commit, or reference in the repo ( <code>master</code> if not specified)
proxy_server	string	(optional) proxy server URL for downloading <code>git</code> or <code>tar</code> charts

## Source Example

```

# type git
---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  release: blog-1
  namespace: default
  wait:
    timeout: 100
    labels:
      component: blog
  source:
    type: git
    location: https://github.com/namespace/repo
    proxy_server: http://my.proxy.server:8888

# type local
---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  release: blog-1
  namespace: default
  wait:
    timeout: 100
  source:
    type: local
    location: /path/to/charts
    subpath: chart
    reference: master

# type tar
---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1

```

(continues on next page)

(continued from previous page)

```

name: blog-1
data:
  release: blog-1
  namespace: default
  wait:
    timeout: 100
  source:
    type: tar
    location: https://localhost:8879/charts/chart-0.1.0.tgz
    subpath: mariadb

```

## Simple Example

```

---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  release: blog-1
  namespace: default
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: blog-1
    reference: new-feat
---
schema: armada/ChartGroup/v2
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
  description: Deploys Simple Service
  chart_group:
    - blog-1
---
schema: armada/Manifest/v2
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - blog-group

```

## Multichart Example

```

---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1

```

(continues on next page)

```
  name: blog-1
data:
  release: blog-1
  namespace: default
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: blog1
    reference: master
---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-2
data:
  release: blog-2
  namespace: default
  source:
    type: tar
    location: https://github.com/namespace/repo/blog2.tgz
    subpath: blog2
---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-3
data:
  release: blog-3
  namespace: default
  source:
    type: local
    location: /home/user/namespace/repo/blog3
---
schema: armada/ChartGroup/v2
metadata:
  schema: metadata/Document/v1
  name: blog-group-1
data:
  description: Deploys Simple Service
  chart_group:
    - blog-2
---
schema: armada/ChartGroup/v2
metadata:
  schema: metadata/Document/v1
  name: blog-group-2
data:
  description: Deploys Simple Service
  chart_group:
    - blog-1
    - blog-3
---
```

(continues on next page)

(continued from previous page)

```

schema: armada/Manifest/v2
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - blog-group-1
    - blog-group-2

```

## Dependency Override Example

```

---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  release: blog-1
  namespace: default
  source:
    type: git
    location: https://github.com/namespace/repo
    subpath: blog-1
    reference: new-feat
  dependencies:
    - blog-dep-1
---
schema: armada/Chart/v2
metadata:
  schema: metadata/Document/v1
  name: blog-1-dep
data:
  release: blog-1-dep
  namespace: default
  source:
    type: git
    location: https://github.com/namespace/dep-repo
    subpath: blog-1-dep
    reference: new-feat
---
schema: armada/ChartGroup/v2
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
  description: Deploys Simple Service
  chart_group:
    - blog-1
---
schema: armada/Manifest/v2

```

(continues on next page)

(continued from previous page)

```
metadata:
  schema: metadata/Document/v1
  name: simple-armada
data:
  release_prefix: armada
  chart_groups:
    - blog-group
```

## References

For working examples please check the examples in our repo [here](#).

## v2 Schemas

Below are the schemas Armada uses to validate Charts, Chart Groups, and Manifests.

### Charts

Charts consist of the smallest building blocks in Armada. A Chart is comparable to a Helm chart. Charts consist of all the labels, dependencies, install and upgrade information, hooks and additional information needed to convey to Helm.

### Chart Groups

A Chart Group consists of a list of charts. Chart Group documents are useful for managing a group of Chart documents together.

### Manifests

A Manifest is the largest building block in Armada. Manifest documents are responsible for managing collections of Chart Group documents.

## Validation Schemas

### Introduction

All schemas below are [Deckhand DataSchema](#) documents, which are essentially JSON schemas, with additional metadata useful for Deckhand to perform [layering](#) and [substitution](#).

The validation schemas below are used by Armada to validate all ingested Charts, Chart Groups, and Manifests. Use the schemas below as models for authoring Armada documents.

### Schemas

- Chart schema.

JSON schema against which all documents with `armada/Chart/v2 metadata.name` are validated.

Listing 4: Schema for `armada/Chart/v2` documents.

```
# JSON schema for validating Armada charts.
---
schema: deckhand/DataSchema/v1
metadata:
```

(continues on next page)

(continued from previous page)

```
name: armada/Chart/v2
schema: metadata/Control/v1
data:
  $schema: http://json-schema.org/schema#
  definitions:
    labels:
      type: object
      additionalProperties:
        type: string
    hook_action:
      type: array
      items:
        properties:
          type:
            type: string
          labels:
            $ref: '#/definitions/labels'
          required:
            - type
          additionalProperties: false
    wait_resource_type_config:
      properties:
        labels:
          $ref: '#/definitions/labels'
        min_ready:
          anyOf:
            - type: integer
            - type: string
          required:
            type: boolean
  type: object
  properties:
    release:
      type: string
    namespace:
      type: string
    values:
      type: object
    # TODO: Remove this, and just read dependencies out of `chart` dir as helm
    # CLI does.
    dependencies:
      type: array
      items:
        type: string
    protected:
      type: object
      properties:
        continue_processing:
          type: boolean
        additionalProperties: false
  test:
    type: object
```

(continues on next page)

```
properties:
  enabled:
    type: boolean
  timeout:
    type: integer
  options:
    type: object
    properties:
      cleanup:
        type: boolean
      additionalProperties: false
  additionalProperties: false
wait:
  type: object
  properties:
    timeout:
      type: integer
  resources:
    anyOf:
      - additionalProperties:
          anyOf:
            - $ref: '#/definitions/wait_resource_type_config'
            - type: array
              items:
                $ref: '#/definitions/wait_resource_type_config'
      - type: array
        items:
          allOf:
            - $ref: '#/definitions/wait_resource_type_config'
            - properties:
                type:
                  type: string
                required:
                  - type

  labels:
    $ref: "#/definitions/labels"
# Config for helm's native '--wait' param.
native:
  type: object
  properties:
    enabled:
      type: boolean
    additionalProperties: false
  additionalProperties: false
source:
  type: object
  properties:
    type:
      type: string
    location:
      type: string
    subpath:
```

(continues on next page)

(continued from previous page)

```

    type: string
  reference:
    type: string
  proxy_server:
    type: string
  auth_method:
    type: string
  required:
    - location
    - type
  delete:
    type: object
    properties:
      timeout:
        type: integer
  upgrade:
    type: object
    properties:
      pre:
        type: object
        additionalProperties: false
        properties:
          delete:
            $ref: '#/definitions/hook_action'
      options:
        type: object
        properties:
          force:
            type: boolean
          no_hooks:
            type: boolean
          additionalProperties: false
        additionalProperties: false
  required:
    - namespace
    - release
    - source
  additionalProperties: false
...

```

This schema is used to sanity-check all Chart documents that are passed to Armada.

- Chart Group schema.

JSON schema against which all documents with `armada/Chart/v2` `metadata.name` are validated.

Listing 5: Schema for `armada/ChartGroup/v2` documents.

```

# JSON schema for validating Armada chart groups.
---
schema: deckhand/DataSchema/v1
metadata:
  name: armada/ChartGroup/v2

```

(continues on next page)

(continued from previous page)

```

  schema: metadata/Control/v1
data:
  $schema: http://json-schema.org/schema#
  properties:
    name:
      type: string
    description:
      type: string
    sequenced:
      type: boolean
    chart_group:
      type: array
      items:
        type: string
  required:
    # TODO: Rename to `charts`?
    - chart_group
  additionalProperties: false
...

```

This schema is used to sanity-check all `Chart Group` documents that are passed to Armada.

- Manifest schema.

JSON schema against which all documents with `armada/Manifest/v2` `metadata.name` are validated.

Listing 6: Schema for `armada/Manifest/v2` documents.

```

# JSON schema for validating Armada manifests.
---
schema: deckhand/DataSchema/v1
metadata:
  name: armada/Manifest/v2
  schema: metadata/Control/v1
data:
  $schema: http://json-schema.org/schema#
  properties:
    release_prefix:
      type: string
    chart_groups:
      type: array
      items:
        type: string
  required:
    - chart_groups
    - release_prefix
  additionalProperties: false
...

```

This schema is used to sanity-check all `Manifest` documents that are passed to Armada.

## Authoring Guidelines

All Armada documents must use the `deckhand/DataSchema/v1` schema.

### v1-v2 Migration

The following migrations must be done when moving from *v1* to *v2* docs.

#### Chart

change	migration
<code>chart_name</code> removed	Remove. It was redundant with <code>metadata.name</code> while at the same time not guaranteeing uniqueness. Log messages now reference <code>metadata.name</code> for improved grep-ability.
<code>test</code> as a boolean removed	<code>test</code> must now be an object.
<code>timeout</code> removed	Use <code>wait.timeout</code> instead.
<code>install</code> removed	Remove. Previously unused.
<code>upgrade.post</code> removed	Remove.
<code>upgrade.pre.update</code> removed	Remove.
<code>upgrade.pre.create</code> removed	Remove.
<code>upgrade.pre.delete[*].name</code> removed	Remove.
<code>upgrade.pre.delete[*]</code> with <code>type: job</code> no longer deletes cronjobs	If you have an item in <code>upgrade.pre.delete</code> and <code>type: job</code> and you also want to delete cronjobs, add another item with <code>type: cronjob</code> and same labels.
<code>upgrade.no_hooks</code> moved to <code>upgrade.options.no_hooks</code> , and now optional	Remove as desired, otherwise move to the new location.
<code>source.subpath</code> now optional, defaults to no subpath.	Remove as desired.
Template naming for template files aligned with Helm CLI.	If a chart was relying on Armada's previous misaligned template naming, where it was omitting the <code>templates/</code> prefix, such as via <code>include</code> argument, that argument will need to be updated. This could also theoretically affect whether the file is ignored, if the old or new name is in <code>.helmignore</code> (unlikely). The fixed template names alone will not cause a release to be updated, as the diff logic accounts for this.
<code>wait</code> improvements	See <i>Wait Improvements</i> .

### Wait Improvements

The *v2 wait API* includes the following changes.

#### Breaking changes

1. `wait.resources` now defaults to all supported resource types, currently `job`, `daemonset`, `statefulset`, `deployment`, and `pod`, with `required` (a new option) set to `false`. The previous default was the equivalent of pods with `required=true`, and jobs with `required=false`.
2. `type: pod` waits now exclude pods owned by other resources, such as controllers, as one should instead wait directly on the controller itself, which per 1. is now the default.
3. Waits are no longer retried due to resources having been modified. This was mildly useful before as an indicator of whether all targeted resources were accounted for, but with 1. and 2. above, we are now tracking top-level resources directly included in the release, rather than generated resources, such as controller-owned pods, so there is no need to wait for them to come into existence.
4. `wait.native.enabled` is now disabled by default. With the above changes, this is no longer useful as a backup mechanism. Having both enabled leads to ambiguity in which wait would fail in each case. More importantly, this must be disabled in order to use the `min_ready` functionality, otherwise helm will wait for 100% anyway. So this prevents accidentally leaving it enabled in that case. Also when the helm native wait times out, this caused the release to be marked `FAILED` by helm, which caused it to be purged and re-installed (unless protected), even though the wait criteria may have eventually succeeded, which is already validated by armada on a retry.

#### New features

##### Per-resource-type overrides

`wait.resources` can now be a dict, mapping individual resource types to wait configurations (or lists thereof), such that one can keep the default configuration for the other resource types, and also disable a given resource type, by mapping it to `false`.

The ability to provide the entire explicit list for `wait.resources` remains in place as well.

##### required

A `required` field is also exposed for items/values in `wait.resources`.

##### allow\_async\_updates

An `allow_async_updates` field is added to `daemonset` and `statefulset` type items/values in `wait.resources`.

#### ChartGroup

change	migration
<code>test_charts</code> removed	Use the Chart schema's <i>test.enabled</i> instead.

#### Manifest

No changes.

## Configuring Armada

Armada uses an INI-like standard `oslo_config` file. A sample file can be generated via `tox`

```
$ tox -e genconfig
$ tox -e genpolicy
```

Customize your configuration based on the information below

## Keystone Integration

Armada requires a service account to use for validating API tokens

### Note

If you do not have a keystone already deploy, then armada can deploy a keystone services:

```
$ armada apply keystone-manifest.yaml
```

```
$ openstack domain create 'ucp'
$ openstack project create --domain 'ucp' 'service'
$ openstack user create --domain ucp --project service --project-domain 'ucp' --password_
↪armada armada
$ openstack role add --project-domain ucp --user-domain ucp --user armada --project_
↪service admin

# OR

$ ./tools/keystone-account.sh
```

The service account must then be included in the `armada.conf`

```
[keystone_authtoken]
auth_type = password
auth_uri = https://<keystone-api>:5000/
auth_url = https://<keystone-api>:35357/
auth_version = 3
delay_auth_decision = true
password = armada
project_domain_name = ucp
project_name = service
user_domain_name = ucp
user_name = armada
```

## Checking Logs

In order to check the logs the logs file will be in `~/armada` directory.

When running Armada in the container you can execute `docker logs` to retrieve logs

```
docker logs [container-name | container-id]
```

### Errors/Exceptions

A guide for interpreting errors/exceptions can be found [here](#).

### Working with SSL

You might run into SSL error with armada if you are not using the correct versions of SSL.

Debugging Checklist:

1. `python -c "import ssl; print ssl.OPENSSL_VERSION"`

If the version that appears is less than 1.0, then problems will occur. Please update to current or use our docker container solve this issue

2. check your urllib3 version, you could run into urllib3 issues. older versions of this lib can cause SSL errors run `pip install --upgrade urllib3` and it should solve this issue

### Issue

If the issue that you are having does not appear here please check the Armada issues on [GitHub](#). If the issue does not exist, please create an issue.

## Armada Install & Usage Guide

### Prerequisites

Kubernetes Cluster

[Helm](#)

*Armada documents*

#### **i** Note

Need to have provided a storage system prior(ceph, nfs)

### Usage

#### **i** Note

The apply command performs two main actions: installing and updating define charts in the armada manifest

1. Pull or Build the Armada Docker Images:

Pull:

```
docker pull quay.io/airshipit/armada:latest-ubuntu_bionic
```

Build:

```
git clone https://opendev.org/airship/armada.git && cd armada/  
docker build . -t quay.io/airshipit/armada:latest-ubuntu_bionic
```

2. Running Armada

- a. docker container

**Note**

Make sure to mount your kubeconfig into `/armada/.kube/config` in the container

**Note**

To run your custom Armada.yamls you need to mount them into the container as shown below. This example is using `examples/` directory in armada [repo](#).

```
docker run -d --net host -p 8000:8000 --name armada -v $(pwd)/etc:/etc/ -v ~/.kube:/armada/.kube/ -v $(pwd)/examples:/examples quay.io/airshipit/armada:latest-ubuntu_
↳bionic
docker exec armada armada --help
```

## b. Helm Install

**Note**

To install Armada via the Helm chart please make sure to provide a Keystone endpoint

```
make charts
```

```
helm install <registry>/armada --name armada --namespace armada
```

## 3. Deploy armada yamls

```
docker exec armada armada apply /examples/openstack-helm.yaml [ --debug ]
```

4. Upgrading charts: modify the armada yaml or chart source code and run `armada apply` above

```
docker exec armada armada apply /examples/openstack-helm.yaml [ --debug ]
```

## 5. Testing Releases:

```
docker exec armada armada test --release=armada-keystone
```

OR

```
docker exec armada armada test --file=/examples/openstack-helm.yaml
```

**Overriding Manifest Values**

It is possible to override manifest values from the command line using the `-set` and `-values` flags. When using the `set` flag, the document type should be specified first, with the target values following in this manner:

```
armada apply --set [ document_type ]:[ document_name ]:[ data_value ]=[ value ]
```

Example:

(continues on next page)

(continued from previous page)

```
armada apply --set chart:blog-1:release="new-blog"  
armada apply --set chart:blog-1:values.blog.new="welcome"
```

**Note**

When overriding values using the set flag, new values will be inserted if they do not exist. An error will only occur if the correct pattern is not used.

There are three types of override types that can be specified: - chart - chart\_group - manifest

An example of overriding the location of a chart:

```
armada apply --set chart:[ chart_name ]:source.location=test [ FILE ]
```

Example:

```
armada apply --set chart:blog-1:release=test [ FILE ]
```

An example of overriding the description of a chart group:

```
armada apply --set chart_group:[ chart_group_name ]:description=test [ FILE ]
```

Example:

```
armada apply examples/simple.yaml --set chart_group:blog-group:description=test
```

An example of overriding the release prefix of a manifest:

```
armada apply --set manifest:[ manifest_name ]:release_prefix=[ value ] [ FILE ]
```

Example:

```
armada apply example/simple.yaml --set manifest:simple-armada:release_prefix=armada-2
```

**Note**

The `--set` flag can be used multiple times.

It is also possible to override manifest values using values specified in a yaml file using the `--values` flag. When using the `--values` flag, a path to the yaml file should be specified in this format:

```
armada apply --values [ path_to_yaml ] [ FILE ]
```

Example:

```
armada apply examples/simple.yaml --values examples/simple-ovr-values.yaml
```

**Note**

The `--values` flag, like the `--set` flag, can be specified more than once. The `--set` and `--values` flag can also be specified at the same time; however, overrides specified by the `--set` flag take precedence over those specified by the `--values` flag.

When creating a yaml file of override values, it should be the same as creating an armada manifest overriding documents with the same schema and metadata name for example:

```

---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-1
data:
  release: chart-example
  namespace: blog-blog
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: blog-2
data:
  release: chart-example-2
  namespace: blog-blog
---
schema: armada/ChartGroup/v1
metadata:
  schema: metadata/Document/v1
  name: blog-group
data:
  description: Change value deploy
  chart_group:
    - blog-1

```

### User bearer token

It is possible to pass the user bearer token from the armada CLI to interact with a kubernetes cluster that has been configured with an external Auth-backend like openstack-keystone.

Example:

```
armada apply --bearer-token [ TOKEN ] --values [ path_to_yaml ] [ FILE ]
```

#### Note

The bearer token option is available for the following commands

armada apply,

### Metrics

Armada exposes metric data, for consumption by [Prometheus](#).

### Exporting

Metric data can be exported via:

- API: Prometheus exporter in the `/metrics` endpoint. The Armada chart includes the appropriate Prometheus scrape configurations for this endpoint.
- CLI: `-metrics-output=<path>` of `apply` command. The `node exporter text file collector` can then be used to export the produced text files to Prometheus.

### Metric Names

Metric names are as follows:

`armada_ + <action> + _ + <metric>`

### Supported <action>s

The below tree of <action>s are measured. Supported prometheus labels are noted. Labels are inherited by sub-actions except as noted.

- `apply`:
  - description: apply a manifest
  - labels: `manifest`
  - sub-actions:
    - \* `chart_handle`:
      - description: fully handle a chart (see below sub-actions)
      - labels:
        - `chart`
        - `action` (install|upgrade|noop) (not included in sub-actions)
        - sub-actions:
          - `chart_download`
          - `chart_deploy`
          - `chart_test`
    - \* `chart_delete`:
      - description: delete a chart (e.g. due to `FAILED` status)
      - labels: `chart`

### Supported <metric>s

- `failure_total`: total failed attempts
- `attempt_total`: total attempts
- `attempt_inprogress`: total attempts in progress

- *duration\_seconds*: duration of each attempt

## Timeouts

The *chart\_handle* and *chart\_test* actions additionally include the following metrics:

- *timeout\_duration\_seconds*: configured chart timeout duration in seconds
- *timeout\_usage\_ratio*:  $= \text{duration\_seconds} / \text{timeout\_duration\_seconds}$

These can help identify charts whose timeouts may need to be changed to avoid potential failures or to achieve faster failures.

## Chart concurrency

The *chart\_handle* action additionally includes the following metric:

- *concurrency\_count*: count of charts being handled concurrently

This can help identify opportunities for greater chart concurrency.

## Exceptions Guide

### Armada Exceptions

#### API Exceptions

**exception** `armada.exceptions.api_exceptions.ApiException`(*message=None, \*\*kwargs*)

Bases: `ArmadaBaseException`

Base class for API exceptions and error handling.

**Troubleshoot:** *Coming Soon*

**message** = 'An unknown API error occurred.'

**exception** `armada.exceptions.api_exceptions.ClientError`(*message=None, \*\*kwargs*)

Bases: `ApiException`

Exception that occurs when the server returns a 500 Internal Server error.

**Troubleshoot:** *Coming Soon*

**message** = 'There was an error listing the Helm chart releases.'

**exception** `armada.exceptions.api_exceptions.ClientForbiddenError`(*message=None, \*\*kwargs*)

Bases: `ApiException`

Exception that occurs when the server returns a 403 Forbidden error.

**Troubleshoot:** *Coming Soon*

**message** = 'There was an error listing the Helm chart releases.'

**exception** `armada.exceptions.api_exceptions.ClientUnauthorizedError`(*message=None, \*\*kwargs*)

Bases: `ApiException`

Exception that occurs when the server returns a 401 Unauthorized error.

**Troubleshoot:** *Coming Soon*

**message** = 'There was an error listing the Helm chart releases.'

### Armada Exceptions

**exception** `armada.exceptions.armada_exceptions.ArmadaTimeoutException(reason)`

Bases: `ArmadaException`

Exception that occurs when Armada times out while processing.

**exception** `armada.exceptions.armada_exceptions.ProtectedReleaseException(release_id, status)`

Bases: `ArmadaException`

Exception that occurs when Armada encounters a release with status other than DEPLOYED that is designated *protected* in the Chart and *continue\_processing* is False.

**exception** `armada.exceptions.armada_exceptions.InvalidValuesYamlException(chart_description)`

Bases: `ArmadaException`

Exception that occurs when Armada encounters invalid values.yaml content in a helm chart.

**exception** `armada.exceptions.armada_exceptions.InvalidOverrideValuesYamlException(chart_description)`

Bases: `ArmadaException`

Exception that occurs when Armada encounters invalid override yaml in helm chart.

**exception** `armada.exceptions.armada_exceptions.ChartDeployException(chart_names)`

Bases: `ArmadaException`

Exception that occurs while deploying charts.

**exception** `armada.exceptions.armada_exceptions.WaitException(message)`

Bases: `ArmadaException`

Exception that occurs while waiting for resources to become ready.

**exception** `armada.exceptions.armada_exceptions.DeploymentLikelyPendingException(release_id, status, last_deployment_age, timeout)`

Bases: `ArmadaException`

Exception that occurs when it is detected that an existing release operation (e.g. install, update, delete) is likely still pending.

**exception** `armada.exceptions.armada_exceptions.PreUpdateJobDeleteException(name, namespace)`

Bases: `ArmadaException`

Exception that occurs when a job deletion.

**Troubleshoot:** *Coming Soon*

### Base Exceptions

**exception** `armada.exceptions.base_exception.ActionForbidden(title=None, description=None, error_list=None, info_list=None, status=None, retry=False)`

Bases: `ArmadaAPIException`

Exception thrown when an action is forbidden.

**Troubleshoot:** *Coming Soon*

**code:** `int` | `None`

An internal application code that a user can reference when requesting support for the error.

**description:** `str` | `None`

Description of the error to send to the client.

**headers:** `HeaderArg` | `None`

Extra headers to add to the response.

**link:** `Link` | `None`

An href that the client can provide to the user for getting help.

**message =** `'Insufficient privilege to perform action.'`

**status:** `ResponseStatus`

HTTP status code or line (e.g., `'200 OK'`).

This may be set to a member of `http.HTTPStatus`, an HTTP status line string or byte string (e.g., `'200 OK'`), or an `int`.

**title:** `str`

Error title to send to the client.

Derived from the `status` if not provided.

## Chartbuilder Exceptions

**exception** `armada.exceptions.chartbuilder_exceptions.HelmChartBuildException`(*chart\_name*,  
*details*)

Bases: `ChartBuilderException`

Exception that occurs when Helm Chart fails to build.

## Kubernetes Exceptions

**exception** `armada.exceptions.k8s_exceptions.KubernetesErrorEventException`(*message=*`None`,  
*\*\*kwargs*)

Bases: `KubernetesException`

Exception for getting an error from the Kubernetes API

**message =** `'An error event was returned from the streaming API.'`

**exception** `armada.exceptions.k8s_exceptions.KubernetesUnknownStreamingEventTypeException`(*message=*`None`,  
*\*\*kwargs*)

Bases: `KubernetesException`

Exception for getting an unknown event type from the Kubernetes API

**message =** `'An unknown event type was returned from the streaming API.'`

**exception** `armada.exceptions.k8s_exceptions.KubernetesWatchTimeoutException`(*message=*`None`,  
*\*\*kwargs*)

Bases: `KubernetesException`

Exception for timing out during a watch on a Kubernetes object

**message =** `'Kubernetes Watch has timed out.'`

## Manifest Exceptions

**exception** `armada.exceptions.manifest_exceptions.ManifestException`(*message=None, \*\*kwargs*)

Bases: `ArmadaBaseException`

An exception occurred while attempting to build an Armada manifest. The exception will return with details as to why.

**Troubleshoot:** *Coming Soon*

`message = 'An error occurred while generating the manifest: %(details)s.'`

## Override Exceptions

**exception** `armada.exceptions.override_exceptions.InvalidOverrideFileException`(*filename*)

Bases: `OverrideException`

Exception that occurs when an invalid override file is provided.

**Troubleshoot:** *Coming Soon*

**exception** `armada.exceptions.override_exceptions.InvalidOverrideValueException`(*override\_command*)

Bases: `OverrideException`

Exception that occurs when an invalid value is used with the set flag.

**Troubleshoot:** *Coming Soon*

**exception** `armada.exceptions.override_exceptions.UnknownDocumentOverrideException`(*doc\_type, doc\_name*)

Bases: `OverrideException`

Exception that occurs when an invalid value is used with the set flag.

**Troubleshoot:** *Coming Soon*

## Source Exceptions

**exception** `armada.exceptions.source_exceptions.ChartSourceException`(*chart\_name, source\_type*)

Bases: `SourceException`

Exception for unknown chart source type.

**Troubleshoot:** *Coming Soon*

**exception** `armada.exceptions.source_exceptions.GitException`(*location*)

Bases: `SourceException`

Exception when an error occurs cloning a Git repository.

**Troubleshoot:** *Coming Soon*

**exception** `armada.exceptions.source_exceptions.InvalidPathException`(*path*)

Bases: `SourceException`

Exception that occurs when a nonexistant path is accessed.

**Troubleshoot:** *Coming Soon*

**exception** `armada.exceptions.source_exceptions.TarballDownloadException`(*tarball\_url*)

Bases: `SourceException`

Exception that occurs when the tarball cannot be downloaded from the provided URL.

**Troubleshoot:** *Coming Soon*

**exception** `armada.exceptions.source_exceptions.TarballExtractException`(*tarball\_dir*)

Bases: `SourceException`

Exception that occurs when extracting the tarball fails.

**Troubleshoot:** *Coming Soon*

## Lint (Validate) Exceptions

**exception** `armada.exceptions.validate_exceptions.InvalidManifestException`(*message=None, \*\*kwargs*)

Bases: `ValidateException`

Exception for invalid manifests.

**Troubleshoot:** *Coming Soon*

**message = 'Armada manifest(s) failed validation. Details: %(error\_messages)s.'**

**exception** `armada.exceptions.validate_exceptions.InvalidChartDefinitionException`(*message=None, \*\*kwargs*)

Bases: `ValidateException`

Exception when invalid chart definition is encountered.

**message = 'Invalid chart definition. Chart definition must be array.'**

**exception** `armada.exceptions.validate_exceptions.InvalidReleaseException`(*message=None, \*\*kwargs*)

Bases: `ValidateException`

Exception that occurs when a release is invalid.

**message = 'Release needs to be a string.'**

**exception** `armada.exceptions.validate_exceptions.InvalidArmadaObjectException`(*message=None, \*\*kwargs*)

Bases: `ValidateException`

Exception that occurs when an Armada object is not declared.

**Troubleshoot:** *Coming Soon*

**message = 'An Armada object failed internal validation. Details: %(details)s.'**

## Armada Plugin

The armada plugin extends all the functionality of Armada to be used as a plugin with Helm.

### Install Plugin

#### Install directly from the repository

```
helm plugin install https://opendev.org/airship/armada.git
```

#### Clone and install locally

```
git clone https://opendev.org/airship/armada.git ~/.helm/plugins/  
helm plugin install ~/.helm/plugins/armada
```

### Usage

#### helm <Name> <action> [options]

```
helm armada apply ~/.helm/plugins/armada/examples/simple.yaml
```

### Sample Configuration File

The following is a sample Armada configuration for adaptation and use. It is auto-generated from Armada when this documentation is built, so if you are having issues with an option, please compare your version of Armada with the version of this documentation.

The sample configuration can also be viewed in [file form](#).

```
[DEFAULT]  
  
#  
# From oslo.log  
#  
  
# If set to true, the logging level will be set to DEBUG instead of the default  
# INFO level. (boolean value)  
# Note: This option can be changed without restarting.  
#debug = false  
  
# The name of a logging configuration file. This file is appended to any  
# existing logging configuration files. For details about logging configuration  
# files, see the Python logging module documentation. Note that when logging  
# configuration files are used then all logging configuration is set in the  
# configuration file and other logging configuration options are ignored (for  
# example, log-date-format). (string value)  
# Note: This option can be changed without restarting.  
# Deprecated group/name - [DEFAULT]/log_config  
#log_config_append = <None>  
  
# Defines the format string for %(asctime)s in log records. Default:  
# %(default)s . This option is ignored if log_config_append is set. (string  
# value)  
#log_date_format = %Y-%m-%d %H:%M:%S  
  
# (Optional) Name of log file to send logging output to. If no default is set,  
# logging will go to stderr as defined by use_stderr. This option is ignored if  
# log_config_append is set. (string value)
```

(continues on next page)

(continued from previous page)

```
# Deprecated group/name - [DEFAULT]/logfile
#log_file = <None>

# (Optional) The base directory used for relative log_file paths. This option
# is ignored if log_config_append is set. (string value)
# Deprecated group/name - [DEFAULT]/logdir
#log_dir = <None>

# DEPRECATED: Uses logging handler designed to watch file system. When log file
# is moved or removed this handler will open a new log file with specified path
# instantaneously. It makes sense only if log_file option is specified and
# Linux platform is used. This option is ignored if log_config_append is set.
# (boolean value)
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: This function is known to have bene broken for long time, and depends
# on the unmaintained library
#watch_log_file = false

# Use syslog for logging. Existing syslog format is DEPRECATED and will be
# changed later to honor RFC5424. This option is ignored if log_config_append
# is set. (boolean value)
#use_syslog = false

# Enable journald for logging. If running in a systemd environment you may wish
# to enable journal support. Doing so will use the journal native protocol
# which includes structured metadata in addition to log messages.This option is
# ignored if log_config_append is set. (boolean value)
#use_journal = false

# Syslog facility to receive log lines. This option is ignored if
# log_config_append is set. (string value)
#syslog_log_facility = LOG_USER

# Use JSON formatting for logging. This option is ignored if log_config_append
# is set. (boolean value)
#use_json = false

# Log output to standard error. This option is ignored if log_config_append is
# set. (boolean value)
#use_stderr = false

# (Optional) Set the 'color' key according to log levels. This option takes
# effect only when logging to stderr or stdout is used. This option is ignored
# if log_config_append is set. (boolean value)
#log_color = false

# The amount of time before the log files are rotated. This option is ignored
# unless log_rotation_type is set to "interval". (integer value)
#log_rotate_interval = 1

# Rotation interval type. The time of the last file change (or the time when
```

(continues on next page)

(continued from previous page)

```

# the service was started) is used when scheduling the next rotation. (string
# value)
# Possible values:
# Seconds - <No description provided>
# Minutes - <No description provided>
# Hours - <No description provided>
# Days - <No description provided>
# Weekday - <No description provided>
# Midnight - <No description provided>
#log_rotate_interval_type = days

# Maximum number of rotated log files. (integer value)
#max_logfile_count = 30

# Log file maximum size in MB. This option is ignored if "log_rotation_type" is
# not set to "size". (integer value)
#max_logfile_size_mb = 200

# Log rotation type. (string value)
# Possible values:
# interval - Rotate logs at predefined time intervals.
# size - Rotate logs once they reach a predefined size.
# none - Do not rotate log files.
#log_rotation_type = none

# Format string to use for log messages with context. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_context_format_string = %(asctime)s.%(msecs)03d %(process)d %(levelname)s
↳ %(name)s [% (global_request_id)s %(request_id)s %(user_identity)s] %(instance)s
↳ %(message)s

# Format string to use for log messages when context is undefined. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_default_format_string = %(asctime)s.%(msecs)03d %(process)d %(levelname)s
↳ %(name)s [-] %(instance)s%(message)s

# Additional data to append to log message when logging level for the message
# is DEBUG. Used by oslo_log.formatters.ContextFormatter (string value)
#logging_debug_format_suffix = %(funcName)s %(pathname)s:%(lineno)d

# Prefix each line of exception output with this format. Used by
# oslo_log.formatters.ContextFormatter (string value)
#logging_exception_prefix = %(asctime)s.%(msecs)03d %(process)d ERROR %(name)s
↳ %(instance)s

# Defines the format string for %(user_identity)s that is used in
# logging_context_format_string. Used by oslo_log.formatters.ContextFormatter
# (string value)
#logging_user_identity_format = %(user)s %(project)s %(domain)s %(system_scope)s %(user_
↳ domain)s %(project_domain)s

# List of package logging levels in logger=LEVEL pairs. This option is ignored

```

(continues on next page)

(continued from previous page)

```

# if log_config_append is set. (list value)
#default_log_levels = amqp=WARN,amqplib=WARN,boto=WARN,qpidd=WARN,sqlalchemy=WARN,
↳suds=INFO,oslo.messaging=INFO,oslo_messaging=INFO,iso8601=WARN,requests.packages.
↳urllib3.connectionpool=WARN,urllib3.connectionpool=WARN,websocket=WARN,requests.
↳packages.urllib3.util.retry=WARN,urllib3.util.retry=WARN,keystonemiddleware=WARN,
↳routes.middleware=WARN,stevedore=WARN,taskflow=WARN,keystoneauth=WARN,oslo.cache=INFO,
↳oslo_policy=INFO,dogpile.core.dogpile=INFO

# Enables or disables publication of error events. (boolean value)
#publish_errors = false

# The format for an instance that is passed with the log message. (string
# value)
#instance_format = "[instance: %(uuid)s] "

# The format for an instance UUID that is passed with the log message. (string
# value)
#instance_uuid_format = "[instance: %(uuid)s] "

# Interval, number of seconds, of log rate limiting. (integer value)
#rate_limit_interval = 0

# Maximum number of logged messages per rate_limit_interval. (integer value)
#rate_limit_burst = 0

# Log level name used by rate limiting. Logs with level greater or equal to
# rate_limit_except_level are not filtered. An empty string means that all
# levels are filtered. (string value)
# Possible values:
# CRITICAL - <No description provided>
# ERROR - <No description provided>
# INFO - <No description provided>
# WARNING - <No description provided>
# DEBUG - <No description provided>
# " - <No description provided>
#rate_limit_except_level = CRITICAL

# Enables or disables fatal status of deprecations. (boolean value)
#fatal_deprecations = false

[cors]

#
# From oslo.middleware
#

# Indicate whether this resource may be shared with the domain received in the
# requests "origin" header. Format: "<protocol>://<host>[:<port>]", no trailing
# slash. Example: https://horizon.example.com (list value)
#allowed_origin = <None>

```

(continues on next page)

(continued from previous page)

```
# Indicate that the actual request can include user credentials (boolean value)
#allow_credentials = true

# Indicate which headers are safe to expose to the API. Defaults to HTTP Simple
# Headers. (list value)
#expose_headers =

# Maximum cache age of CORS preflight requests. (integer value)
#max_age = 3600

# Indicate which methods can be used during the actual request. (list value)
#allow_methods = OPTIONS,GET,HEAD,POST,PUT,DELETE,TRACE,PATCH

# Indicate which header field names may be used during the actual request.
# (list value)
#allow_headers =

[healthcheck]

#
# From oslo.middleware
#
# DEPRECATED: The path to respond to healthcheck requests on. (string value)
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
#path = /healthcheck

# Show more detailed information as part of the response. Security note:
# Enabling this option may expose sensitive details about the service being
# monitored. Be sure to verify that it will not violate your security policies.
# (boolean value)
#detailed = false

# Additional backends that can perform health checks and report that
# information back as part of a request. (list value)
#backends =

# A list of network addresses to limit source ip allowed to access healthcheck
# information. Any request from ip outside of these network addresses are
# ignored. (list value)
#allowed_source_ranges =

# Ignore requests with proxy headers. (boolean value)
#ignore_proxied_requests = false

# Check the presence of a file to determine if an application is running on a
# port. Used by DisableByFileHealthcheck plugin. (string value)
#disable_by_file_path = <None>

# Check the presence of a file based on a port to determine if an application
```

(continues on next page)

(continued from previous page)

```
# is running on a port. Expects a "port:path" list of strings. Used by
# DisableByFilesPortsHealthcheck plugin. (list value)
#disable_by_file_paths =

# Check the presence of files. Used by EnableByFilesHealthcheck plugin. (list
# value)
#enable_by_file_paths =

[keystone_auth_token]

#
# From keystonemiddleware.auth_token
#

# Complete "public" Identity API endpoint. This endpoint should not be an
# "admin" endpoint, as it should be accessible by all end users.
# Unauthenticated clients are redirected to this endpoint to authenticate.
# Although this endpoint should ideally be unversioned, client support in the
# wild varies. If you're using a versioned v2 endpoint here, then this should
# *not* be the same endpoint the service user utilizes for validating tokens,
# because normal end users may not be able to reach that endpoint. (string
# value)
# Deprecated group/name - [keystone_auth_token]/auth_uri
#www_authenticate_uri = <None>

# DEPRECATED: Complete "public" Identity API endpoint. This endpoint should not
# be an "admin" endpoint, as it should be accessible by all end users.
# Unauthenticated clients are redirected to this endpoint to authenticate.
# Although this endpoint should ideally be unversioned, client support in the
# wild varies. If you're using a versioned v2 endpoint here, then this should
# *not* be the same endpoint the service user utilizes for validating tokens,
# because normal end users may not be able to reach that endpoint. This option
# is deprecated in favor of www_authenticate_uri and will be removed in the S
# release. (string value)
# This option is deprecated for removal since Queens.
# Its value may be silently ignored in the future.
# Reason: The auth_uri option is deprecated in favor of www_authenticate_uri
# and will be removed in the S release.
#auth_uri = <None>

# API version of the Identity API endpoint. (string value)
#auth_version = <None>

# Interface to use for the Identity API endpoint. Valid values are "public",
# "internal" (default) or "admin". (string value)
#interface = internal

# Do not handle authorization requests within the middleware, but delegate the
# authorization decision to downstream WSGI components. (boolean value)
#delay_auth_decision = false
```

(continues on next page)

(continued from previous page)

```
# Request timeout value for communicating with Identity API server. (integer
# value)
#http_connect_timeout = <None>

# How many times are we trying to reconnect when communicating with Identity
# API Server. (integer value)
#http_request_max_retries = 3

# Request environment key where the Swift cache object is stored. When
# auth_token middleware is deployed with a Swift cache, use this option to have
# the middleware share a caching backend with swift. Otherwise, use the
# ``memcached_servers`` option instead. (string value)
#cache = <None>

# Required if identity server requires client certificate (string value)
#certfile = <None>

# Required if identity server requires client certificate (string value)
#keyfile = <None>

# A PEM encoded Certificate Authority to use when verifying HTTPS connections.
# Defaults to system CAs. (string value)
#cafile = <None>

# Verify HTTPS connections. (boolean value)
#insecure = false

# The region in which the identity server can be found. (string value)
#region_name = <None>

# Optionally specify a list of memcached server(s) to use for caching. If left
# undefined, tokens will instead be cached in-process. (list value)
# Deprecated group/name - [keystone_auth_token]/memcache_servers
#memcached_servers = <None>

# In order to prevent excessive effort spent validating tokens, the middleware
# caches previously-seen tokens for a configurable duration (in seconds). Set
# to -1 to disable caching completely. (integer value)
#token_cache_time = 300

# (Optional) If defined, indicate whether token data should be authenticated or
# authenticated and encrypted. If MAC, token data is authenticated (with HMAC)
# in the cache. If ENCRYPT, token data is encrypted and authenticated in the
# cache. If the value is not one of these options or empty, auth_token will
# raise an exception on initialization. (string value)
# Possible values:
# None - <No description provided>
# MAC - <No description provided>
# ENCRYPT - <No description provided>
#memcache_security_strategy = None

# (Optional, mandatory if memcache_security_strategy is defined) This string is
```

(continues on next page)

(continued from previous page)

```
# used for key derivation. (string value)
#memcache_secret_key = <None>

# (Optional) Number of seconds memcached server is considered dead before it is
# tried again. (integer value)
#memcache_pool_dead_retry = 300

# (Optional) Maximum total number of open connections to every memcached
# server. (integer value)
#memcache_pool_maxsize = 10

# (Optional) Socket timeout in seconds for communicating with a memcached
# server. (integer value)
#memcache_pool_socket_timeout = 3

# (Optional) Number of seconds a connection to memcached is held unused in the
# pool before it is closed. (integer value)
#memcache_pool_unused_timeout = 60

# (Optional) Number of seconds that an operation will wait to get a memcached
# client connection from the pool. (integer value)
#memcache_pool_conn_get_timeout = 10

# (Optional) Use the advanced (eventlet safe) memcached client pool. (boolean
# value)
#memcache_use_advanced_pool = true

# (Optional) Indicate whether to set the X-Service-Catalog header. If False,
# middleware will not ask for service catalog on token validation and will not
# set the X-Service-Catalog header. (boolean value)
#include_service_catalog = true

# Used to control the use and type of token binding. Can be set to: "disabled"
# to not check token binding. "permissive" (default) to validate binding
# information if the bind type is of a form known to the server and ignore it
# if not. "strict" like "permissive" but if the bind type is unknown the token
# will be rejected. "required" any form of token binding is needed to be
# allowed. Finally the name of a binding method that must be present in tokens.
# (string value)
#enforce_token_bind = permissive

# A choice of roles that must be present in a service token. Service tokens are
# allowed to request that an expired token can be used and so this check should
# tightly control that only actual services should be sending this token. Roles
# here are applied as an ANY check so any role in this list must be present.
# For backwards compatibility reasons this currently only affects the
# allow_expired check. (list value)
#service_token_roles = service

# For backwards compatibility reasons we must let valid service tokens pass
# that don't pass the service_token_roles check as valid. Setting this true
# will become the default in a future release and should be enabled if
```

(continues on next page)

(continued from previous page)

```
# possible. (boolean value)
#service_token_roles_required = false

# The name or type of the service as it appears in the service catalog. This is
# used to validate tokens that have restricted access rules. (string value)
#service_type = <None>

# Enable the SASL(Simple Authentication and Security Layer) if the SASL_enable
# is true, else disable. (boolean value)
#memcache_sasl_enabled = false

# the user name for the SASL (string value)
#memcache_username =

# the username password for SASL (string value)
#memcache_password =

# Authentication type to load (string value)
# Deprecated group/name - [keystone_authtoken]/auth_plugin
#auth_type = <None>

# Config Section from which to load plugin specific options (string value)
#auth_section = <None>

[oslo_middleware]

#
# From oslo.middleware
#

# The maximum body size for each request, in bytes. (integer value)
# Deprecated group/name - [DEFAULT]/osapi_max_request_body_size
# Deprecated group/name - [DEFAULT]/max_request_body_size
#max_request_body_size = 114688

# Whether the application is behind a proxy or not. This determines if the
# middleware should parse the headers or not. (boolean value)
#enable_proxy_headers_parsing = false

# HTTP basic auth password file. (string value)
#http_basic_auth_user_file = /etc/htpasswd

[oslo_policy]

#
# From oslo.policy
#

# DEPRECATED: This option controls whether or not to enforce scope when
# evaluating policies. If ``True``, the scope of the token used in the request
```

(continues on next page)

(continued from previous page)

```
# is compared to the `scope_types` of the policy being enforced. If the
# scopes do not match, an `InvalidScope` exception will be raised. If
# `False`, a message will be logged informing operators that policies are
# being invoked with mismatching scope. (boolean value)
# This option is deprecated for removal.
# Its value may be silently ignored in the future.
# Reason: This configuration was added temporarily to facilitate a smooth
# transition to the new RBAC. OpenStack will always enforce scope checks. This
# configuration option is deprecated and will be removed in the 2025.2 cycle.
#enforce_scope = true

# This option controls whether or not to use old deprecated defaults when
# evaluating policies. If `True`, the old deprecated defaults are not going
# to be evaluated. This means if any existing token is allowed for old defaults
# but is disallowed for new defaults, it will be disallowed. It is encouraged
# to enable this flag along with the `enforce_scope` flag so that you can get
# the benefits of new defaults and `scope_type` together. If `False`, the
# deprecated policy check string is logically OR'd with the new policy check
# string, allowing for a graceful upgrade experience between releases with new
# policies, which is the default behavior. (boolean value)
#enforce_new_defaults = true

# The relative or absolute path of a file that maps roles to permissions for a
# given service. Relative paths must be specified in relation to the
# configuration file setting this option. (string value)
#policy_file = policy.yaml

# Default rule. Enforced when a requested rule is not found. (string value)
#policy_default_rule = default

# Directories where policy configuration files are stored. They can be relative
# to any directory in the search path defined by the config_dir option, or
# absolute paths. The file defined by policy_file must exist for these
# directories to be searched. Missing or empty directories are ignored. (multi
# valued)
#policy_dirs = policy.d

# Content Type to send and receive data for REST based policy check (string
# value)
# Possible values:
# application/x-www-form-urlencoded - <No description provided>
# application/json - <No description provided>
#remote_content_type = application/x-www-form-urlencoded

# server identity verification for REST based policy check (boolean value)
#remote_ssl_verify_server_cert = false

# Absolute path to ca cert file for REST based policy check (string value)
#remote_ssl_ca_cert_file = <None>

# Absolute path to client cert for REST based policy check (string value)
#remote_ssl_client_cert_file = <None>
```

(continues on next page)

(continued from previous page)

```
# Absolute path client key file REST based policy check (string value)
#remote_ssl_client_key_file = <None>

# Timeout in seconds for REST based policy check (floating point value)
# Minimum value: 0
#remote_timeout = 60
```

### Sample Policy File

The following is a sample Armada policy file for adaptation and use. It is auto-generated from Armada when this documentation is built, so if you are having issues with an option, please compare your version of Armada with the version of this documentation.

The sample policy file can also be viewed in [file form](#).

## 1.5.3 Commands Guide

### Armada - Apply

#### Commands

```
Usage: armada apply [OPTIONS] [LOCATIONS]...
```

This **command** installs and updates charts defined **in** Armada manifest.

The apply argument must be relative path to Armada Manifest. Executing apply **command** once will install all charts defined **in** manifest. Re-executing apply **command** will execute upgrade.

To see how to create an Armada manifest: <https://airship-armada.readthedocs.io/en/latest/operations/>

To install or upgrade charts, run:

```
$ armada apply examples/simple.yaml
```

To override a specific value **in** a Manifest, run:

```
$ armada apply examples/simple.yaml --set manifest:simple-armada:release=
↪ "wordpress"
```

Or to override several values **in** a Manifest, reference a values.yaml-formatted file:

```
$ armada apply examples/simple.yaml --values examples/simple-ovr-values.yaml
```

Options:

```
--api                Contacts service endpoint.
--disable-update-post Disable post-update Helm operations.
```

(continues on next page)

(continued from previous page)

<code>--disable-update-pre</code>	Disable pre-update Helm operations.
<code>--enable-chart-cleanup</code>	Clean up unmanaged charts.
<code>--metrics-output TEXT</code>	The output path <b>for</b> metric data
<code>--use-doc-ref</code>	Use armada manifest file reference.
<code>--set TEXT</code>	Use to override Armada Manifest values. Accepts overrides that adhere to the format <path>:<to>:<property>=<value> to specify a primitive or <path>:<to>:<property>=<value1>,...,<valueN> to specify a list of values.
<code>--timeout INTEGER</code>	Specifies <b>time</b> to <b>wait for</b> each chart to fully finish deploying.
<code>-f, --values TEXT</code>	Use to override multiple Armada Manifest values by reading overrides from a values.yaml-type file.
<code>--wait</code>	Force Helm to <b>wait until</b> all charts are deployed, rather than using each charts specified <b>wait</b> policy. This is equivalent to sequenced chartgroups.
<code>--target-manifest TEXT</code>	The target manifest to run. Required <b>for</b> specifying which manifest to run when multiple are available.
<code>--bearer-token TEXT</code>	User Bearer token
<code>--debug</code>	Enable debug logging.
<code>--help</code>	Show this message and exit.

## Synopsis

The apply command will consume an armada manifest which contains group of charts that it will deploy via the Helm CLI into your Kubernetes cluster. Executing the `armada apply` again on existing armada deployment will start an update of the armada deployed charts.

```
armada apply armada-manifest.yaml [--debug]
```

If you remove `armada/Charts/v1` from the `armada/ChartGroups/v1` in the armada manifest and execute an `armada apply` with the `--enable-chart-cleanup` flag. Armada will remove undefined releases with the armada manifest's `release_prefix` keyword.

## Armada - Test

### Commands

```
Usage: armada test [OPTIONS]
```

This **command** `test` deployed charts

The **test** **command** will run the release chart tests either via a manifest or by targeting a release.

To obtain armada deployed releases:

```
$ armada test --file examples/simple.yaml
```

(continues on next page)

(continued from previous page)

```
To test release:

    $ armada test --release blog-1

Options:
--cleanup           Delete test pods after test completion
--enable-all       Run disabled chart tests
--file TEXT         armada manifest
--release TEXT      helm release
--target-manifest TEXT The target manifest to run. Required for
                    specifying which manifest to run when multiple
                    are available.
--help             Show this message and exit.
```

### Synopsis

The test command will perform helm test defined on the release. Test command can test a single release or a manifest.

### Armada - Validate

#### Commands

```
Usage: armada validate [OPTIONS] FILENAME

This command validates Armada Manifest

The validate argument must be a relative path to Armada manifest

    $ armada validate examples/simple.yaml

Options:
--help Show this message and exit.
```

### Synopsis

The validate command will take in an Armada manifest and will validate if it is correctly defined and consumable.

## INDEX

### A

ActionForbidden, 48  
ApiException, 47  
ArmadaTimeoutException, 48

### C

ChartDeployException, 48  
ChartSourceException, 50  
ClientError, 47  
ClientForbiddenError, 47  
ClientUnauthorizedError, 47  
code (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 48

### D

DeploymentLikelyPendingException, 48  
description (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 49

### G

GitException, 50

### H

headers (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 49  
HelmChartBuildException, 49

### I

InvalidArmadaObjectException, 51  
InvalidChartDefinitionException, 51  
InvalidManifestException, 51  
InvalidOverrideFileException, 50  
InvalidOverrideValueException, 50  
InvalidOverrideValuesYamlException, 48  
InvalidPathException, 50  
InvalidReleaseException, 51  
InvalidValuesYamlException, 48

### K

KubernetesErrorEventException, 49

KubernetesUnknownStreamingEventTypeException,  
49

KubernetesWatchTimeoutException, 49

### L

link (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 49

### M

ManifestException, 50  
message (*armada.exceptions.api\_exceptions.ApiException*  
attribute), 47  
message (*armada.exceptions.api\_exceptions.ClientError*  
attribute), 47  
message (*armada.exceptions.api\_exceptions.ClientForbiddenError*  
attribute), 47  
message (*armada.exceptions.api\_exceptions.ClientUnauthorizedError*  
attribute), 47  
message (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 49  
message (*armada.exceptions.k8s\_exceptions.KubernetesErrorEventExcepti*  
attribute), 49  
message (*armada.exceptions.k8s\_exceptions.KubernetesUnknownStreamin*  
attribute), 49  
message (*armada.exceptions.k8s\_exceptions.KubernetesWatchTimeoutExc*  
attribute), 49  
message (*armada.exceptions.manifest\_exceptions.ManifestException*  
attribute), 50  
message (*armada.exceptions.validate\_exceptions.InvalidArmadaObjectExc*  
attribute), 51  
message (*armada.exceptions.validate\_exceptions.InvalidChartDefinitionEx*  
attribute), 51  
message (*armada.exceptions.validate\_exceptions.InvalidManifestException*  
attribute), 51  
message (*armada.exceptions.validate\_exceptions.InvalidReleaseException*  
attribute), 51

### P

PreUpdateJobDeleteException, 48  
ProtectedReleaseException, 48

## S

status (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 49

## T

TarballDownloadException, 50

TarballExtractException, 51

title (*armada.exceptions.base\_exception.ActionForbidden*  
attribute), 49

## U

UnknownDocumentOverrideException, 50

## W

WaitException, 48