pegleg

Release 0.1.0

May 16, 2018

Contents

Tip: The Undercloud Platform is part of the AIC CP (AT&T Integrated Cloud Containerized Platform). More details may be found by using the Treasuremap

Use of `tox -e docs` will build an HTML version of this documentation that can be viewed using a browser at docs/build/index.html on the local filesystem.

CHAPTER 1

Conventions and Standards

1.1 Getting Started

1.1.1 What is Pegleg?

Pegleg is a document aggregator that will aggregate all the documents in a repository and pack them into a single YAML file. This allows for operators to structure their site definitions in a maintainable directory layout, while providing them with the automation and tooling needed to aggregate, lint, and render those documents for deployment.

For more information on the documents that Pegleg works on see Document Fundamentals.

1.1.2 Basic Usage

Before using Pegleg, you must:

Clone the Pegleg repository

```
git clone https://github.com/att-comdev/pegleg
```

and install the required packages in pegleg/src/bin/pegleg

```
pip3 install -r pegleg/src/bin/pegleg/requirements.txt -r pegleg/src/bin/pegleg/test-
↪requirements.txt
```

Next, add your repos containing your site definition libraries into your local system where Pegleg is running, as Pegleg can only work on files available in the local directory.

You will then be able to use all of Pegleg's features through the CLI. See CLI for more information.

1.2 Document Fundamentals

The definition of a site consists of a set of small YAML documents that are managed by Deckhand. Each document is identified by a `schema` top-level key and the `metadata.name` value that uniquely identifies a particular document of the type `schema`. Deckhand provides functionality allowing documents to be authored such that data from multiple documents can be merged.

- Abstact vs Concrete - Documents define a value in `metadata.layeringDefinition.abstract` to determine if a document is abstract (a value of `true`) or concrete (a value of `false`). When calling the `/revisions/{id}/rendered-documents` API, only concrete documents are returned.
- Layering - Document layering is used for whole documents that have known defaults but may need to be transformed in specific instances.
- Substitution - Data substitution is used for extracting particular values from a document's data section (whole or in-part) and inserting that data into a destination document (at the root of the data section or deeper into a document).

1.3 Shared Documents

1.3.1 Secrets

Several generic document types exist to support storing sensitive data encrypted.

These must be utilized for all data considered sensitive.

1.3.2 Global Catalogue Documents

Deckhand's layering functionality can be utilized in several ways, i.e site definitions. At the `global` layer there will be several documents providing different configurations for an object or service. Each of these will be abstract documents. They can be incorporated into a particular site definition by creating a concrete child document in the `site` layer that selects the correct `global` parent. The child can then do further customization on the configuration if needed.

As a best practice, `global` level documents using the catalog pattern should utilize the layering labels `component` and `configuration` to provide a consistent method for children documents select the correct parent. The below example shows a set of documents for two configuration options for OpenStack Keystone: one using local SQL-backed identity stores and one using an LDAP backend. A site definition can then select and customize the appropriate option.

When using a catalogue document, it is important to review that document to ensure you understand all the requirements for it.

- Abstract documents are not required to be fully formed, so selecting a catalogue document may require the child document to add data so the document passes validation. In the below example, the child document adds several required fields to the catalogue Chart: `chart_name`, `release`, and `namespace`.
- A catalogue document may define substitutions with the expectation that the substitution source documents are defined at a lower layer. In the example below, all of the required credentials in the chart are defined as substitutions in the `global` catalogue document, but the source documents for the substitutions are defined in the `site` layer.

This catalogue pattern can also be utilized for the `type` layer if needed.

1.3.3 Global Layer

```
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: ldap-backed-keystone
  labels:
    component: keystone
    configuration: ldap-backed
  layeringDefinition:
    abstract: true
    layer: global
  storagePolicy: cleartext
  substitutions:
    - src:
        schema: deckhand/Passphrase/v1
        name: keystone_admin_password
        path: .
      dest:
        path: .values.endpoints.identity.auth.admin.password
    - src:
        schema: deckhand/Passphrase/v1
        name: mariadb_admin_password
        path: .
      dest:
        path: .values.endpoints.oslo_db.auth.admin.password
    - src:
        schema: deckhand/Passphrase/v1
        name: mariadb_keystone_password
        path: .
      dest:
        path: .values.endpoints.oslo_db.auth.user.password
    - src:
        schema: pegleg/SoftwareVersions/v1
        name: software-versions
        path: .charts.ucp.keystone
      dest:
        path: .source
    - src:
        schema: pegleg/StringValue/v1
        name: ldap_userid
        src: .
      dest:
        path: .values.conf.ks_domains.cicd.identity.ldap.user
        pattern: '(^USERID)'
    - src:
        schema: deckhand/Passphrase/v1
        name: ldap_userid_password
        path: .
      dest:
        path: .values.conf.ks_domain.cicd.identity.ldap.password
data:
  install:
    no_hooks: false
  upgrade:
    no_hooks: false
  pre:
```

```
    delete:
      - type: job
        labels:
          job-name: keystone-db-sync
      - type: job
        labels:
          job-name: keystone-db-init
  post:
    delete: []
    create: []
  values:
    conf:
      keystone:
        identity:
          driver: sql
          default_domain_id: default
          domain_specific_drivers_enabled: True
          domain_configurations_from_database: True
          domain_config_dir: /etc/keystonedomains
      ks_domains:
        cicd:
          identity:
            driver: ldap
            ldap:
              url: "ldap://your-ldap-server.example.com"
              user: "USERID@example.com"
              password: USERID_PASSWORD_REPLACEME
              suffix: "dc=example,dc=com"
              query_scope: sub
              page_size: 1000
              user_tree_dn: "DC=example,DC=com"
              user_objectclass: user
              user_name_attribute: sAMAccountName
              user_mail_attribute: mail
              user_enabled_attribute: userAccountControl
              user_enabled_mask: 2
              user_enabled_default: 512
              user_attribute_ignore: "default_project_id,tenants,projects,password"
    replicas: 2
    labels:
      node_selector_key: ucp-control-plane
      node_selector_value: enabled
...
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: sql-backed-keystone
  labels:
    component: keystone
    configuration: sql-backed
  layeringDefinition:
    abstract: true
    layer: global
  substitutions:
    - src:
        schema: deckhand/Passphrase/v1
        name: keystone_admin_password
```

```
        path: .
      dest:
        path: .values.endpoints.identity.auth.admin.password
    - src:
        schema: deckhand/Passphrase/v1
        name: mariadb_admin_password
        path: .
      dest:
        path: .values.endpoints.oslo_db.auth.admin.password
    - src:
        schema: deckhand/Passphrase/v1
        name: mariadb_keystone_password
        path: .
      dest:
        path: .values.endpoints.oslo_db.auth.user.password
    - src:
        schema: pegleg/SoftwareVersions/v1
        name: software-versions
        path: .charts.ucp.keystone
      dest:
        path: .source
data:
  timeout: 300
  install:
    no_hooks: false
  upgrade:
    no_hooks: false
    pre:
      delete:
        - name: keystone-bootstrap
          type: job
          labels:
            application: keystone
            component: bootstrap
        - name: keystone-credential-setup
          type: job
          labels:
            application: keystone
            component: credential-setup
        - name: keystone-db-init
          type: job
          labels:
            application: keystone
            component: db-init
        - name: keystone-db-sync
          type: job
          labels:
            application: keystone
            component: db-sync
        - name: keystone-fernet-setup
          type: job
          labels:
            application: keystone
            component: fernet-setup
  values: {}
  source: {}
...
```

1.3.4 Site Layer

```
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: ucp-helm-toolkit
  layeringDefinition:
    abstract: false
    layer: site
  substitutions:
    - src:
        schema: pegleg/SoftwareVersions/v1
        name: software-versions
        path: .charts.ucp.helm-toolkit
      dest:
        path: .source
data:
  chart_name: ucp-helm-toolkit
  release: ucp-helm-toolkit
  namespace: ucp
  timeout: 100
  values: {}
  source: {}
  dependencies: []
...
---
schema: armada/Chart/v1
metadata:
  schema: metadata/Document/v1
  name: ucp-keystone
  layeringDefinition:
    abstract: false
    layer: site
    parentSelector:
      component: keystone
      configuration: ldap-backed
    actions:
      - method: merge
        path: .
data:
  chart_name: ucp-keystone
  release: ucp-keystone
  namespace: ucp
  dependencies:
    - ucp-helm-toolkit
...
---
schema: deckhand/Passphrase/v1
metadata:
  schema: metadata/Document/v1
  name: ldap_userid_password
  storagePolicy: encrypted
data: a-secret-password
...
---
schema: deckhand/Passphrase/v1
metadata:
```

```
  schema: metadata/Document/v1
  name: keystone_admin_password
  storagePolicy: encrypted
data: a-secret-password
...
---
schema: deckhand/Passphrase/v1
metadata:
  schema: metadata/Document/v1
  name: mariadb_admin_password
  storagePolicy: encrypted
data: a-secret-password
...
---
schema: deckhand/Passphrase/v1
metadata:
  schema: metadata/Document/v1
  name: mariadb_keystone_password
  storagePolicy: encrypted
data: a-secret-password
...
---
schema: pegleg/StringValue/v1
metadata:
  schema: metadata/Document/v1
  name: keystone_ldap_userid
  storagePolicy: cleartext
data: myuser
...
```

1.4 Definition Artifact Layout

The definition artifacts are stored in the below directory structure. This structure is used only to assist humans in maintaining the data. When the documents are consumed by the UCP services, they are viewed as a flat set of all documents.:

```
deployment_files/deployment_files
 |- /global
 |     |- /common
 |     |     |- {definition library}
 |     |- /v1.0
 |           |- {definition library}
 |- /type
 |     |- /production
 |     |     |- /v1.0
 |     |           |- {definition library}
 |     |- /cicd
 |     |     |- /v1.0
 |     |           |- {definition library}
 |     |- /labs
 |           |- /v1.0
 |                 |- {definition library}
 |- /site
       |- /{sitename}
             |- site_definition.yaml
```

```
          |- {definition library}
```

The root-level listings of `global`, `type` and `site` are the layers as listed in the Deckhand _LayeringPolicy http://deckhand.readthedocs.io/en/latest/layering.html document. The process of choosing the definition libraries to compose the actual design for a site is described below.

1.4.1 site_definition.yaml

The site_definition.yaml file is what selects the definition libraries to use for a site. Additional metadata can be added to this file as needed to meet requirements.:

```
---
schema: pegleg/SiteDefinition/v1
metadata:
  layeringDefinition:
    abstract: false
    layer: 'site'
  name: 'mtn13b.1'
  schema: metadata/Document/v1
  storagePolicy: cleartext
data:
  platform_name: 'integration'
  revision: 'v1.0'
  site_type: 'cicd'
```

The `revision` field is used to select the definition libraries in the `global` layer. This layer will be composed of a union of documents in the `common` definition library and the definition library for the `revision`. The `revision` field and the `site_type` fields select the definition library from the `type` layer. And the `site` layer is defined by the single defintion library under the sitename.

1.5 Definition Library Layout

The definition library layout is replicated in each location that the site definition contains a set of documents.:

```
  {library root}
    |- /schemas
    |    |- /{namespace}
    |        |- /{kind}
    |            |- {version}.yaml
    |
    |- /profiles
    |    |- /hardware
    |    |- /host
    |
    |- /pki
    |    |- kubernetes-nodes.yaml
    |
    |- /secrets
    |    |- /certifcate-authorities
    |    |- /certificates
    |    |- /keypairs
    |    |- /passphrases
    |
    |- /software
```

```
    |     |- /charts
    |     |     |- /{chart collection}
    |     |     |     |- dependencies.yaml
    |     |     |     |- /{chartgroup}
    |     |     |           |- chart-group.yaml
    |     |     |           |- {chart1}.yaml
    |     |     |           |- {chart2}.yaml
    |     |     |
    |     |     |- /{chart collection}
    |     |           |- dependencies.yaml
    |     |           |- /{chartgroup}
    |     |                 |- chart-group.yaml
    |     |                 |- {chart1}.yaml
    |     |                 |- {chart2}.yaml
    |     |
    |     |- /config
    |     |     |- Docker.yaml
    |     |     |- Kubelet.yaml
    |     |     |- versions.yaml
    |     |
    |     |- /manifests
    |           |- bootstrap.yaml
    |           |- site.yaml
    |
    |- /networks
    |     |- /physical
    |     |     |- sitewide.yaml
    |     |     |- rack1.yaml
    |     |
    |     |- KubernetesNetwork.yaml
    |     |- common-addresses.yaml
    |
    |- /baremetal
          |- rack1.yaml
          |- rack2.yaml

* Schemas - The schemas should all be sourced from the UCP
  service repositories. Care should be taken that the schemas
  included in the site definition are taken from the version of
  the service being deployed in the site.
* Software
  * /config/versions.yaml will contain a manifest of all the
    chart, image and package versions. These should be substituted
    into all other documents that define version information.
  * dependencies.yaml - Contains Armada chart definitions that are
    only utilized as dependencies for other charts (e.g. helm-toolkit)
  * Chart collection - Loose organization of chart groups
    such as 'kubernetes', 'ucp', 'osh'
* Physical networks and baremetal nodes can be split into files
  in whatever way makes sense. The best practice here to define
  them by racks is only a suggestion.
```

1.6 Pegleg CLI

The Pegleg CLI is used in conjunction with the script located in pegleg/tools called pegleg.sh.

```
$WORKSPACE = Location of the folder that holds the repositories containing
the site definition libraries. Pegleg makes no assumptions about the root
directory. $WORKSPACE is /workspace in the container context.

    Example: $WORKSPACE=/home/ubuntu/all_repos

$IMAGE = Location of pegleg docker image.

    Example: $IMAGE=quay.io/attcomdev/pegleg:latest
```

To run:

```
export WORKSPACE=<repo_location>
export IMAGE=<docker_image>
./pegleg.sh <command> <options>
```

1.6.1 CLI Options

-v / –verbose

Enable debug logging.

Site

This allows you to set the primary and auxiliary repositories.

-p / –primary

Path to the root of the primary (containing site_definition.yaml) repo. (Required).

-a / –auxiliary

Path to the root of an auxiliary repo.

```
./pegleg.sh site -p <primary_repo> -a <auxiliary_repo> <command> <options>

Example:
./pegleg.sh site -p /workspace/repo_1 -a /workspace/repo_2
<command> <options>
```

Collect

Output complete config for one site. It is assumed that all lint errors have been corrected already.

site_name

Name of the site. (Required).

-s / –save-location

Where to output.

```
./pegleg.sh <command> <options> collect site_name -s save_location

Example:
./pegleg.sh site -p /workspace/repo_1 -a /workspace/repo_2
collect site_name -s /workspace
```

Impacted

Find sites impacted by changed files.

-i / –input

List of impacted files.

-o / –output

Where to output.

```
./pegleg impacted -i <input_stream> -o <output_stream>
```

List

List known sites.

-o/–output

Where to output.

```
./pegleg <command> <options> list

Example:
./pegleg site -p /workspace/repo_1 list -o /workspace
```

Show

Show details for one site.

site_name

Name of site. (Required).

-o /–output

Where to output.

```
./pegleg <command> <options> show site_name

Example:
./pegleg site -p /workspace/repo_1 show site_name -o /workspace
```

Lint

Sanity checks for repository content. Validations for linting are done utilizing Deckhand Validations.

```
./pegleg.sh lint -p <primary_repo> -a <auxiliary_repo>
-f -x <lint_code> -w <lint_code>

Example:

./pegleg.sh lint -p /workspace/site-repo -a /workspace/secondary-repo
-x P001 -x P002 -w P003
```

-p / –primary

Path to the root of the primary (containing site_definition.yaml) repo. (Required).

-a / –auxiliary

Path to the root of an auxiliary repo.

-f / –fail-on-missing-sub-src

Raise Deckhand exception on missing substitution sources. Defaults to True.

-x <code>

Will excluded the specified lint option. -w takes priority over -x.

-w <code>

Will warn of lint failures from the specified lint options.

```
If you expect certain lint failures, then those lint options can be
excluded or you can choose to be warned about those failures using the
codes below.

P001 - Document has storagePolicy cleartext (expected is encrypted) yet
its schema is a mandatory encrypted type.

Where mandatory encrypted schema type is one of:
* deckhand/CertificateAuthorityKey/v1
* deckhand/CertificateKey/v1
* deckhand/Passphrase/v1
* deckhand/PrivateKey/v1

P002 - Deckhand rendering is expected to complete without errors.
P003 - All repos contain expected directories.
```