ironik Documentation

Release 0.1.5

Jonathan Decker

May 16, 2022

CONTENTS:

CHAPTER

ONE

IRONIK

Warning: This tool is still in early development and only the core features are available.

1.1 Features

- Utilize OpenStack and Rancher APIs to automatically deploy Kubernetes cluster
- Customize the configuration using templates
- Install new Kubernetes versions including deploying the external cloud controller manager for OpenStack

1.2 Installation

You can install *ironik* via pip from PyPI:

```
$ pip install ironik
```

Alternatively, *ironik* can also be used as a container to avoid installing it:

```
$ docker run --rm -ti -v $(pwd):/app docker.gitlab.gwdg.de/jonathan.decker1/ironik/
↪cli:latest ironik --help
```

This can be abbreviated using an alias:

```
$ alias dironik='docker run --rm -ti -v $(pwd):/app docker.gitlab.gwdg.de/jonathan.
↪decker1/ironik/cli:latest ironik'
$ dironik --help
```

1.3 Usage

Please see the Usage Instructions for details.

Kubernetes can also be deployed manually on OpenStack and Rancher. See the Manual Deployment Instructions for a full guide.

1.4 TODOs

- Update Code documentation to use Google code doc style
- Improve print messages during execution
- Implement a template validator
- Implement cluster validation
- Set up test suite
- Implement automatic config fetching
- Add functionality for undoing deployments and other helpful commands

1.5 Contributing

Contributions are very welcome. To learn more, see the Contributor Guide.

1.6 Credits

This package was created with cookietemple using Cookiecutter based on Hypermodern_Python_Cookiecutter.

CHAPTER

TWO

INSTALLATION

2.1 Stable release

To install ironik, run this command in your terminal:

```
$ pip install ironik
```

This is the preferred method to install ironik, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

2.2 From sources

The sources for ironik can be downloaded from the Gitlab repo. Please note that you require poetry to be installed.

You can either clone the public repository:

```
$ git clone https://gitlab.gwdg.de/jonathan.decker1/ironik.git
```

Once you have a copy of the source, you can install it with:

```
$ make install
```

CHAPTER

THREE

CLI USAGE

Setting up a Kubernetes cluster on OpenStack and Rancher requires a lot of configuration details. Ironik uses a template file to hold said configurations that must be filled out before anything can be deployed.

First run the following command to receive a blank template:

```
$ ironik template
```

This creates the file *ironik_template.yaml* in the local directory. Open this file with a text editor.

3.1 Filling out the template

The template has multiple sections: - deployment_options - kubernetes_config - network_config - openstack_config - openstack_credentials - rancher_config - rancher_credentials

3.1.1 Deployment options are used to configure the validation of the cluster once it is setup.

This is not implemented yet and can be ignored for now.

3.1.2 Kubernetes config allows setting the number of master and worker nodes to deploy as well as what roles they should have.

Furthermore, the version of Kubernetes can be set here.

master_node_roles lists all the roles the master nodes should have. This can be mostly left as it is unless master nodes should, for example, not have the worker role.

number_master_nodes is the number of master nodes to deploy initially. This can be changed later via Rancher.

number_worker_nodes is the number of worker nodes to deploy initially. This can be changed later via Rancher.

version is the Kubernetes version to be deployed. The version must be available in the Rancher instance used and exactly match the version string there.

worker_node_roles lists all the roles the worker nodes should have. This should usually be left as just *worker*.

3.1.3 Network config describes what ports should be opened in the security group created in OpenStack.

It also allows setting the port range that Kubernetes is allowed to use. These can be left as it is as the list of ports is based on the Rancher documentation and what ports it requires.

required_TCP_ports sets the TCP ports to open in the security group. These should usually be left alone. Additional ports can later be opened in the security group in OpenStack.

required_UDP_ports sets the UDP ports to open in the security group. These should usually be left alone. Additional ports can later be opened in the security group in OpenStack.

worker_port_range_max sets the high end of the ports that Kubernetes can use. This is both set in the OpenStack security group as well as in the Kubernetes settings.

worker_port_range_min sets the low end of the ports that Kubernetes can use. This is both set in the OpenStack security group as well as in the Kubernetes settings.

3.1.4 OpenStack config needs to be filled out for the deployment to work.

default_flavor_name is the flavor to use for the nodes in OpenStack. OpenStack web UI -> Compute -> Instances -> Launch Instance -> Flavor, for example: 'm1.medium'

default_image_name is the image to use for the nodes in OpenStack. OpenStack web UI -> Compute -> Images, for example: 'Ubuntu 20.04.3 Server x86_64 (ssd)'

lb_provider is the load balancer system deployed in OpenStack. OpenStack web UI -> Network -> Load Balancers -> Create a new load balancer and check what is set as its provider. Common *lb_provider* are *amphora*, *octavia* or in older systems *haproxy*.

openstack_auth_url is the authentication endpoint to use for OpenStack. OpenStack web UI -> API access -> Identity, append "/v3" to it.

private_network_id is the private network into which the nodes should be deployed. OpenStack web UI -> Network -> select Networks -> pick a private network -> select overview -> ID

project_domain_name is the domain of the project in OpenStack. OpenStack web UI -> Identity -> Projects -> Domain Name

region_name is the region used by OpenStack. This is usually "RegionOne".

remote_ip_prefix is the IP mask for the IPs used inside the OpenStack network. This can be for example *10.254.1.0/24*.

security_group_name sets the name of the security group in OpenStack to create. If this groups already exist, it will be deleted and recreated.

use_octavia determines whether to use the Octavia load balancer in OpenStack. Should Octavia not be installed, set this to *false*.

user_domain_name is the domain used when logging in to OpenStack. Often its the same as the *project_domain_name*

volume_size sets the size of volumes to be created for each node in OpenStack. If set to 0 no volumes will be created.

volume_type sets the type of volume to use in OpenStack. OpenStack web UI -> Volumes -> Volumes -> Create Volume, see the available types.

3.1.5 Next are the OpenStack credentials.

password is the OpenStack password used to login. *project_id* is the OpenStack project id.

OpenStack web UI -> Identity -> Projects -> Project ID

username is the OpenStack username used to login.

3.1.6 Rancher config set configuration on for Rancher.

engine_install_url is the url from where the docker engine install script should be fetched. This can be left as it is.

new_cluster_admin_user_name is the name of the user who should be admin of the new cluster in Rancher. If the user does not exist, it will be created.

new_cluster_admin_user_password is the password of the user who should be admin of the new cluster. If it is left empty, it will be created. If the user already exist, it will be ignored.

rancher_api_base_url is the url used to access the Rancher API. It is commonly the base web address of the Rancher instance with "/v3" at the end.

ssh_user`is the name of the default ssh user for new nodes in OpenStack. This might be `root or *cloud* depending on the OpenStack instance.

3.1.7 Rancher credentials are the Rancher API credentials.

rancher_access_key is the access key part of the Rancher API key.

rancher_secret_key is the secret key part of the Rancher API key. The Rancher API key must be unscoped and can be created via the Rancher web interface.

3.2 Starting deployment

Once the template has been filled out it can be used to start a deployment.

```
$ ironik deploy cluster_name ./ironik_template.yaml
```

Replace *cluster_name* with the desired name for the new cluster.

Ironik will test out the credentials and check whether enough resources are available in OpenStack. Then it reports said information in a table and asks for confirmation.

After confirming the start of the deployment it will create the node templates in Rancher, the cluster in Rancher and wait for the cluster to deploy.

After the cluster is ready it will attempt to create a new user and print the generated password into the console if a new user was created.

Afterwards Ironik will apply the manifests used to install the OpenStack controller manager and the Cinder driver.

CHAPTER

FOUR

MANUAL KUBERNETES DEPLOYMENT

Instruction set for deploying Kubernetes via Rancher on OpenStack using the out-of tree cloud control manager.

4.1 Preface

Since a few years, Kubernetes is trying to move away from in-tree support for cloud provider to out-of-tree or external support. See https://kubernetes.io/blog/2019/04/17/the-future-of-cloud-providers-in-kubernetes/ This essentially means that instead of containing the code for interacting with a given cloud provider in the main Kubernetes repository, the cloud controller manager takes over this task instead. This controller can then be expanded with a plugin to handle a respective cloud provider. OpenStack in-tree support has recently been deprecated such that future deployments of Kubernetes have to use the out-of-tree approach.

The code for the official OpenStack plugin for the cloud control manager can be found here: https://github.com/kubernetes/cloud-provider-openstack Furthermore, the configurations shown later in this guide are either based on the official OpenStack plugin or on this repo https://github.com/rootsami/terraform-rancher2

These instructions were tested on both Rancher2.5 and Rancher2.6 for Kubernetes 1.20 and 1.23.

4.2 Preparation

This must be done once and can be used to deploy any number of Kubernetes cluster.

Kubectl must be installed for later configuration of the cluster.

4.2.1 (Optional) Setup OpenStack CLI

The OpenStack CLI can be used instead of the Horizon web interface for configuring OpenStack. https://github.com/openstack/python-openstackclient

Install via pip

```
pip install python-openstackclient
```

Set the required variables

```
export OS_AUTH_URL="https://api.prod.cloud.gwdg.de:5000/v3"
export OS_IDENTITY_API_VERSION= "3"
export OS_PROJECT_NAME=
export OS_PROJECT_DOMAIN_NAME= "GWDG"
```

(continues on next page)

(continued from previous page)

```
export OS_USERNAME=
export OS_USER_DOMAIN_NAME= "GWDG"
export OS_PASSWORD=
```

Project name is the name of the OpenStack project that is shown at the top when logged in to OpenStack horizon. Username and password are your username and password used for logging in to OpenStack. If password is not provided, the CLI will query it for every command. When using export to set the password make sure that the machine is secure or that the entry is removed from the shell history.

4.2.2 Enable OpenStack Node Driver

The OpenStack Node Driver must be enabled in Rancher for the node deployment to work.

Rancher2.5

In the web UI -> Tools -> Drivers -> Node Drivers -> Select OpenStack and activate

Rancher2.6

In the web UI -> Open Left sidebar -> Cluster Management -> Drivers -> Node Drivers -> Select OpenStack and activate it

4.2.3 Security group

OpenStack web UI -> Network -> Security Groups -> Create a new group called “k8s-node” -> Manage its rules and add the following rules. CIDR should always be set as 10.254.1.0/24 as this is the internal IP range used by OpenStack for its nodes and it prevents external access. This security group should also be added to the Rancher server hosts. If Rancher is not hosted via OpenStack on this IP range, add all rules again with the CIDR of the Rancher server. Rules:

Protocol	Port(s)
TCP	22 (SSH)
TCP	80 (HTTP)
TCP	443 (HTTPS)
TCP	2376
TCP	2379
TCP	2380
TCP	6443
TCP	6783
TCP	6784
TCP	8443
TCP	8472
TCP	9099
TCP	9100
TCP	9433
TCP	9913
TCP	10248
TCP	10250
TCP	10254
TCP	30000-32767
UDP	6783
UDP	6784
UDP	8443
UDP	8472
UDP	30000-32767

This prevents future headaches by opening all ports that Rancher might need according to its documentation: https://rancher.com/docs/rancher/v2.6/en/installation/requirements/ports/

via CLI

```
openstack security group create k8s-node
openstack security group rule create --remote-ip 10.254.1.0/24 --dst-port PORT --
↪protocol PROTOCOL k8s-node
```

Insert for PORT the number specified above. For port ranges use start:end. For PROTOCOL insert either TCP or UDP as specified in the table.

4.2.4 Node Template

Find the following values in the OpenStack web UI or via the CLI and note them down.

flavorName:

OpenStack web UI -> Compute -> Instances -> Launch Instance -> Flavor -> choose one and note down the name, for example: 'm1.medium'

Or via CLI -> *openstack flavor list*

imageName:

OpenStack web UI -> Compute -> Images -> choose one, for example: 'Ubuntu 20.04.3 Server x86_64 (ssd)'

Or via CLI -> *openstack image list*

netId:

OpenStack web UI -> Network -> select Networks -> pick a private network -> select overview -> ID

Or via CLI -> *openstack network list* -> private network ID

password:

The password for your OpenStack account you use to login.

tenantId:

OpenStack web UI -> Identity -> Projects -> Project ID

Or via CLI -> *openstack project list* -> ID

username:

The username for your OpenStack account you sue to login.

Node template

Rancher2.5

Rancher web UI -> click profile picture in top right -> Node Templates -> Add Template -> OpenStack -> fill the template according to the following template while filling in the values gathered above

Rancher2.6

In the web UI -> Open Left sidebar -> Cluster Management -> REK1 Configuration -> Node Templates -> Add Template -> OpenStack -> fill the template according to the following template while filling in the values gathered above

Node template template

```
authURL: https://api.prod.cloud.gwdg.de:5000/v3
domainName: GWDG
endpointType: publicURL
flavorName:
imageName:
netId:
password:
region: RegionOne
secGroups: k8s-node,default
sshUser: cloud
tenantDomainName: GWDG
tenantId:
username:
volumeName: rancher-volume
volumeSize: 0
volumeType: ssd
```

4.3 Kubernetes Cluster Deployment

Kubernetes Cluster deployment on OpenStack using Rancher.

4.3.1 Cluster Creation

Rancher2.5

Rancher web UI -> Global context -> Add Cluster -> OpenStack

Rancher2.6

Rancher web UI -> Home -> Create -> OpenStack

Set a name for the cluster. Add at least one node pool using the node template created above. Set for at least one node pool control plane and etcd to active. Give the node a fitting name. This will be used as a prefix for nodes of this pool, for example *CLUSTERNAME-master* and *CLUSTERNAME-worker*, where CLUSTERNAME is the name of the Kubernetes cluster you are about to create.

Under Kubernetes Options select the desired Kubernetes Version and under Cloud Provider select External.

Press create. This will create node instance on OpenStack using the Node template and spin up a Kubernetes cluster. This will take some time, usually 10 to 15 min.

When viewed in Rancher, all nodes with be marked with a taint “uninitialized=NoSchedule”.

To finish initialization, the external cloud control manager for OpenStack must be configured and deployed.

4.3.2 OpenStack cloud control manager

The next steps require kubectl to be installed and working.

Rancher2.5

In the Rancher Web UI -> Select the new cluster -> Top right Kubeconfig File

Rancher2.6

In the Rancher Web UI -> Select the new cluster -> Top right Copy or Download Kubeconfig File

Copy these settings to a file called config and place it under *~/.kube/config* on your host. See [[#Managing multiple kubeconfigs]] for how to handle multiple kubeconfigs.

Next gather the details required to fill out the cloud conf file.

username:

Your OpenStack username.

password:

Your OpenStack password.

tenant-id:

Your OpenStack project id, see tenantId above.

subnet-id:

OpenStack Web UI -> Network -> Networks -> private network -> Subnets -> ID

Or via CLI -> *openstack subnet list* -> ID

floating-network-id:

OpenStack Web UI -> Network -> Networks -> public -> Overview -> ID

Or via CLI -> *openstack network list* -> public network ID

router-id:

OpenStack Web UI -> Network -> Routers -> select router -> Overview -> ID

Or via CLI -> *openstack router list* -> ID

Cloud conf

Create a file called cloud.conf and fill it with the template below. Then fill in the open fields as described above.

```
[Global]
auth-url = https://api.prod.cloud.gwdg.de:5000/v3
username =
password =
region = RegionOne
tenant-id =
domain-name = GWDG

[BlockStorage]
ignore-volume-az = true
trust-device-path = false

[Networking]
public-network-name = public

[LoadBalancer]
use-octavia = false
subnet-id =
floating-network-id =
create-monitor = false
manage-security-groups = true
monitor-max-retries = 0
enabled = true
lb-version = v2
lb-provider = haproxy

[Route]
router-id =

[Metadata]
request-timeout = 0
```

Afterwards use kubectl to create a secret from this config:

```
kubectl create secret -n kube-system generic cloud-config --from-file=cloud.conf
```

Deploying the cloud controller manager

Create yaml files with the following contents:

cloud-controller-manager-role-bindings.yaml

```
apiVersion: v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRoleBinding
  metadata:
    name: system:cloud-node-controller
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: system:cloud-node-controller
  subjects:
  - kind: ServiceAccount
    name: cloud-node-controller
    namespace: kube-system
- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRoleBinding
  metadata:
    name: system:pvl-controller
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: system:pvl-controller
  subjects:
  - kind: ServiceAccount
    name: pvl-controller
    namespace: kube-system
- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRoleBinding
  metadata:
    name: system:cloud-controller-manager
  roleRef:
    apiGroup: rbac.authorization.k8s.io
    kind: ClusterRole
    name: system:cloud-controller-manager
  subjects:
  - kind: ServiceAccount
    name: cloud-controller-manager
    namespace: kube-system
kind: List
metadata: {}
```

cloud-controller-manager-roles.yaml

```
apiVersion: v1
items:
- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRole
  metadata:
```

(continues on next page)

(continued from previous page)

```
  name: system:cloud-controller-manager
rules:
- apiGroups:
  - coordination.k8s.io
  resources:
  - leases
  verbs:
  - get
  - create
  - update
- apiGroups:
  - ""
  resources:
  - events
  verbs:
  - create
  - patch
  - update
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - '*'
- apiGroups:
  - ""
  resources:
  - nodes/status
  verbs:
  - patch
- apiGroups:
  - ""
  resources:
  - services
  verbs:
  - list
  - patch
  - update
  - watch
- apiGroups:
  - ""
  resources:
  - serviceaccounts
  verbs:
  - create
  - get
- apiGroups:
  - ""
  resources:
  - serviceaccounts/token
  verbs:
  - create
```

(continues on next page)

(continued from previous page)

```
  - apiGroups:
    - ""
    resources:
    - persistentvolumes
    verbs:
    - '*'
  - apiGroups:
    - ""
    resources:
    - endpoints
    verbs:
    - create
    - get
    - list
    - watch
    - update
  - apiGroups:
    - ""
    resources:
    - configmaps
    verbs:
    - get
    - list
    - watch
  - apiGroups:
    - ""
    resources:
    - secrets
    verbs:
    - list
    - get
    - watch
- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRole
  metadata:
    name: system:cloud-node-controller
  rules:
  - apiGroups:
    - ""
    resources:
    - nodes
    verbs:
    - '*'
  - apiGroups:
    - ""
    resources:
    - nodes/status
    verbs:
    - patch
  - apiGroups:
    - ""
    resources:
```

(continues on next page)

(continued from previous page)

```
    - events
    verbs:
    - create
    - patch
    - update
- apiVersion: rbac.authorization.k8s.io/v1
  kind: ClusterRole
  metadata:
    name: system:pvl-controller
  rules:
  - apiGroups:
    - ""
    resources:
    - persistentvolumes
    verbs:
    - '*'
  - apiGroups:
    - ""
    resources:
    - events
    verbs:
    - create
    - patch
    - update
kind: List
metadata: {}
```

openstack-cloud-controller-manager-ds.yaml

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: cloud-controller-manager
  namespace: kube-system
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: openstack-cloud-controller-manager
  namespace: kube-system
  labels:
    k8s-app: openstack-cloud-controller-manager
spec:
  selector:
    matchLabels:
      k8s-app: openstack-cloud-controller-manager
  updateStrategy:
    type: RollingUpdate
  template:
    metadata:
      labels:
        k8s-app: openstack-cloud-controller-manager
```

(continues on next page)

(continued from previous page)

```
    spec:
      nodeSelector:
        node-role.kubernetes.io/controlplane: "true"
      securityContext:
        runAsUser: 1001
      tolerations:
      - key: node.cloudprovider.kubernetes.io/uninitialized
        value: "true"
        effect: NoSchedule
      - key: node-role.kubernetes.io/controlplane
        effect: NoSchedule
        value: "true"
      - key: node-role.kubernetes.io/etcd
        effect: NoExecute
        value: "true"
      serviceAccountName: cloud-controller-manager
      containers:
        - name: openstack-cloud-controller-manager
          image: docker.io/k8scloudprovider/openstack-cloud-controller-manager:latest
          args:
            - /bin/openstack-cloud-controller-manager
            - --v=1
            - --cloud-config=$(CLOUD_CONFIG)
            - --cloud-provider=openstack
            - --use-service-account-credentials=true
            - --bind-address=127.0.0.1
            - --cluster-cidr=10.254.1.0/24
          volumeMounts:
            - mountPath: /etc/kubernetes/pki
              name: k8s-certs
              readOnly: true
            - mountPath: /etc/ssl/certs
              name: ca-certs
              readOnly: true
            - mountPath: /etc/config
              name: cloud-config-volume
              readOnly: true
          resources:
            requests:
              cpu: 200m
          env:
            - name: CLOUD_CONFIG
              value: /etc/config/cloud.conf
      hostNetwork: true
      volumes:
      - hostPath:
          path: /etc/kubernetes/pki
          type: DirectoryOrCreate
        name: k8s-certs
      - hostPath:
          path: /etc/ssl/certs
          type: DirectoryOrCreate
```

(continues on next page)

(continued from previous page)

```
        name: ca-certs
      - name: cloud-config-volume
        secret:
          secretName: cloud-config
```

Notice that the third file sets the flag *–cluster-cidr=10.254.1.0/24*. If used in a different ip range, this must be updated.

Apply the three files using kubectl

```
kubectl apply -f cloud-controller-manager-roles.yaml
kubectl apply -f cloud-controller-manager-role-bindings.yaml
kubectl apply -f openstack-cloud-controller-manager-ds.yaml
```

This should after a short time update the nodes as shown in Rancher to become initialized and be ready for deployment. This further configures how to deploy load balancer using the GWDG OpenStack deployment and its rather outdated load balancer system.

Cinder CSI Driver

To enable Kubernetes to satisfy PVC using Cinder volumes, the cinder csi plugin must be installed.

Create another yaml file for this.

cinder-csi-plugin.yaml

```
# This YAML file contains RBAC API objects,
# which are necessary to run csi controller plugin

apiVersion: v1
kind: ServiceAccount
metadata:
  name: csi-cinder-controller-sa
  namespace: kube-system

---
# external attacher
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-attacher-role
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "patch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["csinodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["volumeattachments"]
    verbs: ["get", "list", "watch", "patch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["volumeattachments/status"]
    verbs: ["patch"]
```

(continues on next page)

(continued from previous page)

```
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-attacher-binding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-controller-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: csi-attacher-role
  apiGroup: rbac.authorization.k8s.io

---
# external Provisioner
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-provisioner-role
rules:
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "create", "delete"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch", "update"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["storageclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["nodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["storage.k8s.io"]
    resources: ["csinodes"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list", "watch", "create", "update", "patch"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshots"]
    verbs: ["get", "list"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotcontents"]
    verbs: ["get", "list"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
```

(continues on next page)

(continued from previous page)

```
  name: csi-provisioner-binding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-controller-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: csi-provisioner-role
  apiGroup: rbac.authorization.k8s.io

---
# external snapshotter
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-snapshotter-role
rules:
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list", "watch", "create", "update", "patch"]
  # Secret permission is optional.
  # Enable it if your driver needs secret.
  # For example, `csi.storage.k8s.io/snapshotter-secret-name` is set in␣
↪VolumeSnapshotClass.
  # See https://kubernetes-csi.github.io/docs/secrets-and-credentials.html for more␣
↪details.
  #  - apiGroups: [""]
  #    resources: ["secrets"]
  #    verbs: ["get", "list"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotclasses"]
    verbs: ["get", "list", "watch"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotcontents"]
    verbs: ["create", "get", "list", "watch", "update", "delete"]
  - apiGroups: ["snapshot.storage.k8s.io"]
    resources: ["volumesnapshotcontents/status"]
    verbs: ["update"]
---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-snapshotter-binding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-controller-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: csi-snapshotter-role
  apiGroup: rbac.authorization.k8s.io
---
```

(continues on next page)

(continued from previous page)

```
# External Resizer
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-resizer-role
rules:
  # The following rule should be uncommented for plugins that require secrets
  # for provisioning.
  # - apiGroups: [""]
  #   resources: ["secrets"]
  #   verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["persistentvolumes"]
    verbs: ["get", "list", "watch", "patch"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["pods"]
    verbs: ["get", "list", "watch"]
  - apiGroups: [""]
    resources: ["persistentvolumeclaims/status"]
    verbs: ["update", "patch"]
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["list", "watch", "create", "update", "patch"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-resizer-binding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-controller-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: csi-resizer-role
  apiGroup: rbac.authorization.k8s.io

---
kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  namespace: kube-system
  name: external-resizer-cfg
rules:
- apiGroups: ["coordination.k8s.io"]
  resources: ["leases"]
  verbs: ["get", "watch", "list", "delete", "update", "create"]
```

(continues on next page)

(continued from previous page)

```
---
kind: RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-resizer-role-cfg
  namespace: kube-system
subjects:
  - kind: ServiceAccount
    name: csi-cinder-controller-sa
    namespace: kube-system
roleRef:
  kind: Role
  name: external-resizer-cfg
  apiGroup: rbac.authorization.k8s.io

---
# This YAML file contains CSI Controller Plugin Sidecars
# external-attacher, external-provisioner, external-snapshotter
# external-resize, liveness-probe

kind: Service
apiVersion: v1
metadata:
  name: csi-cinder-controller-service
  namespace: kube-system
  labels:
    app: csi-cinder-controllerplugin
spec:
  selector:
    app: csi-cinder-controllerplugin
  ports:
    - name: dummy
      port: 12345

---
kind: StatefulSet
apiVersion: apps/v1
metadata:
  name: csi-cinder-controllerplugin
  namespace: kube-system
spec:
  serviceName: "csi-cinder-controller-service"
  replicas: 1
  selector:
    matchLabels:
      app: csi-cinder-controllerplugin
  template:
    metadata:
      labels:
        app: csi-cinder-controllerplugin
    spec:
```

(continues on next page)

(continued from previous page)

```
      serviceAccount: csi-cinder-controller-sa
      containers:
        - name: csi-attacher
          image: k8s.gcr.io/sig-storage/csi-attacher:v3.1.0
          args:
            - "--csi-address=$(ADDRESS)"
            - "--timeout=3m"
          env:
            - name: ADDRESS
              value: /var/lib/csi/sockets/pluginproxy/csi.sock
          imagePullPolicy: "IfNotPresent"
          volumeMounts:
            - name: socket-dir
              mountPath: /var/lib/csi/sockets/pluginproxy/
        - name: csi-provisioner
          image: k8s.gcr.io/sig-storage/csi-provisioner:v2.1.1
          args:
            - "--csi-address=$(ADDRESS)"
            - "--timeout=3m"
            - "--default-fstype=ext4"
            - "--feature-gates=Topology=true"
            - "--extra-create-metadata"
          env:
            - name: ADDRESS
              value: /var/lib/csi/sockets/pluginproxy/csi.sock
          imagePullPolicy: "IfNotPresent"
          volumeMounts:
            - name: socket-dir
              mountPath: /var/lib/csi/sockets/pluginproxy/
        - name: csi-snapshotter
          image: k8s.gcr.io/sig-storage/csi-snapshotter:v2.1.3
          args:
            - "--csi-address=$(ADDRESS)"
            - "--timeout=3m"
          env:
            - name: ADDRESS
              value: /var/lib/csi/sockets/pluginproxy/csi.sock
          imagePullPolicy: Always
          volumeMounts:
            - mountPath: /var/lib/csi/sockets/pluginproxy/
              name: socket-dir
        - name: csi-resizer
          image: k8s.gcr.io/sig-storage/csi-resizer:v1.1.0
          args:
            - "--csi-address=$(ADDRESS)"
            - "--timeout=3m"
            - "--handle-volume-inuse-error=false"
          env:
            - name: ADDRESS
              value: /var/lib/csi/sockets/pluginproxy/csi.sock
          imagePullPolicy: "IfNotPresent"
          volumeMounts:
```

(continues on next page)

(continued from previous page)

```
            - name: socket-dir
              mountPath: /var/lib/csi/sockets/pluginproxy/
          - name: liveness-probe
            image: k8s.gcr.io/sig-storage/livenessprobe:v2.1.0
            args:
              - "--csi-address=$(ADDRESS)"
            env:
              - name: ADDRESS
                value: /var/lib/csi/sockets/pluginproxy/csi.sock
            volumeMounts:
              - mountPath: /var/lib/csi/sockets/pluginproxy/
                name: socket-dir
          - name: cinder-csi-plugin
            image: docker.io/k8scloudprovider/cinder-csi-plugin:latest
            args:
              - /bin/cinder-csi-plugin
              - "--nodeid=$(NODE_ID)"
              - "--endpoint=$(CSI_ENDPOINT)"
              - "--cloud-config=$(CLOUD_CONFIG)"
              - "--cluster=$(CLUSTER_NAME)"
            env:
              - name: NODE_ID
                valueFrom:
                  fieldRef:
                    fieldPath: spec.nodeName
              - name: CSI_ENDPOINT
                value: unix://csi/csi.sock
              - name: CLOUD_CONFIG
                value: /etc/config/cloud.conf
              - name: CLUSTER_NAME
                value: kubernetes
            imagePullPolicy: "IfNotPresent"
            ports:
              - containerPort: 9808
                name: healthz
                protocol: TCP
            # The probe
            livenessProbe:
              failureThreshold: 5
              httpGet:
                path: /healthz
                port: healthz
              initialDelaySeconds: 10
              timeoutSeconds: 10
              periodSeconds: 60
            volumeMounts:
              - name: socket-dir
                mountPath: /csi
              - name: secret-cinderplugin
                mountPath: /etc/config
                readOnly: true
        volumes:
```

(continues on next page)

(continued from previous page)

```
        - name: socket-dir
          emptyDir:
        - name: secret-cinderplugin
          secret:
            secretName: cloud-config
---
# This YAML defines all API objects to create RBAC roles for csi node plugin.

apiVersion: v1
kind: ServiceAccount
metadata:
  name: csi-cinder-node-sa
  namespace: kube-system
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-nodeplugin-role
rules:
  - apiGroups: [""]
    resources: ["events"]
    verbs: ["get", "list", "watch", "create", "update", "patch"]

---
kind: ClusterRoleBinding
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: csi-nodeplugin-binding
subjects:
  - kind: ServiceAccount
    name: csi-cinder-node-sa
    namespace: kube-system
roleRef:
  kind: ClusterRole
  name: csi-nodeplugin-role
  apiGroup: rbac.authorization.k8s.io

---
# This YAML file contains driver-registrar & csi driver nodeplugin API objects,
# which are necessary to run csi nodeplugin for cinder.

kind: DaemonSet
apiVersion: apps/v1
metadata:
  name: csi-cinder-nodeplugin
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app: csi-cinder-nodeplugin
  template:
    metadata:
```

(continues on next page)

(continued from previous page)

```
      labels:
        app: csi-cinder-nodeplugin
    spec:
      tolerations:
        - operator: Exists
      serviceAccount: csi-cinder-node-sa
      hostNetwork: true
      containers:
        - name: node-driver-registrar
          image: k8s.gcr.io/sig-storage/csi-node-driver-registrar:v1.3.0
          args:
            - "--csi-address=$(ADDRESS)"
            - "--kubelet-registration-path=$(DRIVER_REG_SOCK_PATH)"
          lifecycle:
            preStop:
              exec:
                command: ["/bin/sh", "-c", "rm -rf /registration/cinder.csi.openstack.
↪org /registration/cinder.csi.openstack.org-reg.sock"]
          env:
            - name: ADDRESS
              value: /csi/csi.sock
            - name: DRIVER_REG_SOCK_PATH
              value: /var/lib/kubelet/plugins/cinder.csi.openstack.org/csi.sock
            - name: KUBE_NODE_NAME
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
          imagePullPolicy: "IfNotPresent"
          volumeMounts:
            - name: socket-dir
              mountPath: /csi
            - name: registration-dir
              mountPath: /registration
        - name: liveness-probe
          image: k8s.gcr.io/sig-storage/livenessprobe:v2.1.0
          args:
            - --csi-address=/csi/csi.sock
          volumeMounts:
            - name: socket-dir
              mountPath: /csi
        - name: cinder-csi-plugin
          securityContext:
            privileged: true
            capabilities:
              add: ["SYS_ADMIN"]
            allowPrivilegeEscalation: true
          image: docker.io/k8scloudprovider/cinder-csi-plugin:latest
          args:
            - /bin/cinder-csi-plugin
            - "--nodeid=$(NODE_ID)"
            - "--endpoint=$(CSI_ENDPOINT)"
            - "--cloud-config=$(CLOUD_CONFIG)"
```

(continues on next page)

(continued from previous page)

```
        env:
          - name: NODE_ID
            valueFrom:
              fieldRef:
                fieldPath: spec.nodeName
          - name: CSI_ENDPOINT
            value: unix://csi/csi.sock
          - name: CLOUD_CONFIG
            value: /etc/config/cloud.conf
        imagePullPolicy: "IfNotPresent"
        ports:
          - containerPort: 9808
            name: healthz
            protocol: TCP
        # The probe
        livenessProbe:
          failureThreshold: 5
          httpGet:
            path: /healthz
            port: healthz
          initialDelaySeconds: 10
          timeoutSeconds: 3
          periodSeconds: 10
        volumeMounts:
          - name: socket-dir
            mountPath: /csi
          - name: kubelet-dir
            mountPath: /var/lib/kubelet
            mountPropagation: "Bidirectional"
          - name: pods-probe-dir
            mountPath: /dev
            mountPropagation: "HostToContainer"
          - name: secret-cinderplugin
            mountPath: /etc/config
            readOnly: true
    volumes:
      - name: socket-dir
        hostPath:
          path: /var/lib/kubelet/plugins/cinder.csi.openstack.org
          type: DirectoryOrCreate
      - name: registration-dir
        hostPath:
          path: /var/lib/kubelet/plugins_registry/
          type: Directory
      - name: kubelet-dir
        hostPath:
          path: /var/lib/kubelet
          type: Directory
      - name: pods-probe-dir
        hostPath:
          path: /dev
          type: Directory
```

(continues on next page)

(continued from previous page)

```
        - name: secret-cinderplugin
          secret:
            secretName: cloud-config

---
apiVersion: storage.k8s.io/v1
kind: CSIDriver
metadata:
  name: cinder.csi.openstack.org
spec:
  attachRequired: true
  podInfoOnMount: true
  volumeLifecycleModes:
  - Persistent
  - Ephemeral

---
# This YAML file contains StorageClass definition
# and makes it default storageclass

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-sc-cinderplugin
  annotations:
    storageclass.kubernetes.io/is-default-class: "true"
provisioner: cinder.csi.openstack.org
```

And apply it using

```
kubectl apply -f cinder-csi-plugin.yaml
```

This creates a storage class called *csi-sc-cinderplugin* that should be used for creating volumes.

Updating cloud.conf

Should it be necessary to update cloud.conf it can be done like this.

Create or update a file called cloud.conf as described above. Replace the existing secret using this:

```
kubectl create secret generic cloud-config --from-file=cloud.conf --dry-run -n kube-
↪system -o yaml | kubectl apply -n kube-system -f -
```

Next find the pod running the openstack control manager using

```
kubectl get pod -n kube-system
```

and delete it

```
kubectl delete pod openstack-cloud-controller-manager-xxxx
```

xxxx will be some random code that needs to be identified via the first command. This will cause the pod to be recreated and to load the new cloud.conf file.

Should any problem occur, the logs of the openstack-cloud-controller-manager pod can also help troubleshoot the issue.

```
kubectl logs openstack-cloud-controller-manager-xxxx
```

4.4 Validate that it works

The next steps are optional and are just to confirm that the cluster can: - Deploy workloads - Deploy Load balancer that connect to OpenStack - Expose a service - Claim a volume

4.4.1 Deploy a workload

Rancher2.5

Rancher UI -> Global context -> Select your new cluster -> Projects/Namespaces -> Under Project: Default press Add Namespace -> name it 'test' -> Create -> Click Project: Default -> Workloads overview should be open -> Deploy -> name it "test-hello" -> as docker image set "rancher/hello-world" -> Launch

Rancher2.6

Rancher UI -> Home -> Select your new cluster -> Projects/Namespaces -> Under Project: Default press Create Namespace -> Name it "test" -> Create -> Workload -> Create -> Deployment -> Namespace to "test" -> Name the workload "test-hello" -> as docker image set "rancher/hello-world" -> Create

Observe that it should be scheduled after a few seconds.

4.4.2 Deploy a load balancer

Rancher2.5

Select Apps -> Launch -> search for Nginx and select NGINX Ingress Controller -> and click it -> namespace to use existing -> test -> Launch

Click on the nginx app and observe that it is deployed and a load balancer is created in the cluster.

Rancher2.6

Select Apps & Marketplace -> search for Nginx and select NGINX Ingress Controller -> click it -> Install -> Set namespace to test -> Next -> change ingress class to "nginx-test" -> Install

After about a minute the load balancer should also appear in OpenStack. Find the load balancer under OpenStack web UI -> Network -> Load Balancers

Via the CLI -> The OpenStack CLI only supports loadbalancers via octavia, the API can still be accessed via manual requests. See [[#Access loadbalancers outside of the web UI]]

If you currently have no free floating IPs, the load balancer setup will not complete. If the load balancer has not claimed a floating IP, release one: OpenStack web UI -> Network -> Floating IPs -> actions -> release floating IP Be careful not to release the floating IP already claimed by the load balancer.

Via the CLI -> Use *openstack floating ip list* and *openstack floating ip delete ID* to release one floating IP

Once the load balancer in OpenStack is ready and has claimed an IP, the nginx app in Rancher should also be ready.

Make a note of the floating ip that was claimed.

4.4.3 Expose a service

Rancher2.5

From the nginx App overview -> Resources -> Workloads -> Load Balancing -> Observe that a L4 Balancer already exists -> Add Ingress -> name it “test-ingress” -> Set namespace to “test” -> Set path to “/” -> choose “test-hello” as target -> Set port as 80 -> Open Labels & Annotations drop down -> Add annotation with key “kubernetes.io/ingress.class” and value “nginx” -> Save A hostname ending in sslip.io will be generated.

Rancher2.6

First create a service Service Discovery -> Services -> Create -> ClusterIP -> name it “test-service” -> listening port and target port to 80 -> Selectors -> Set key as “workload.user.cattle.io/workloadselector” and value to “apps.deployment-test-test-hello” -> Create -> Observe that “test-hello” appears as a pod for that service Then create an ingress Service Discovery -> Ingresses -> Create -> name it “test-ingress” -> set request host to “http://test-ingress.test.FLOATING-IP.sslip.io”, where FLOATING-IP is the floating IP noted earlier -> set Path prefix to / -> Select “test-service” as target service and set port to 80 -> Labels&Annotations -> Add Annotation -> Set key to “kubernetes.io/ingress.class” and value “nginx-test” -> Create

The hostname should point to a web page with the Rancher logo writing “Hello World!”. It might take a moment for the link to work. Click the link and confirm that the workload is exposed.

Nginx webhook validation workaround

When trying to create the above described ingress, Rancher might refuse with the error:

Internal error occurred: failed calling webhook “validate.nginx.ingress.kubernetes.io”: an error on the server (“”) has prevented the request from succeeding

A simple workaround for this is to run

```
kubectl delete -A ValidatingWebhookConfiguration ingress-nginx-admission
```

This removes the offending validation hook.

4.4.4 Claiming a volume

Rancher2.5

Select Apps -> Launch -> search for mysql and select it -> namespace to use an existing namespace and pick “test” -> activate PVC and ensure it uses the default storage class that is “csi-sc-cinderplugin” -> Launch

Click on the mysql app and observe that it is deployed and a volume claim is created in the cluster.

Rancher2.6

Select Apps&Marketplace -> search for sql and select cockroachdb -> Install -> namespace to "test" -> Next -> Storage per Node to 10Gi -> Install

After a moment the app should deploy and open volume claim(s) that becomes satisfied by volume(s) from OpenStack after a few moments.

In the OpenStack web UI -> Volumes -> Volumes And one volume should have the description "Created by OpenStack Cinder CSI driver"

Or via the CLI -> *openstack volume list*

4.4.5 Cleanup

Delete the namespace test to cleanup all created resources.

Rancher2.5

Rancher UI -> Global context -> Select cluster -> Projects/Namespaces -> check the box next to the test namepsace -> Delete -> Confirm

Rancher2.6

Rancher UI -> Cluster -> Projects/Namespaces -> check the box next to the test namepsace -> Hold CTRL and press Delete

This also removes the load balancer and volumes in OpenStack. CockroachDB might not remove all its volumes, check manually and remove the remaining volumes.

4.5 Appendix

4.5.1 Managing multiple kubeconfigs

Use a tool such as kubectx to manage what context to load. Option 1: One large config Append any new config to the config file. Option 2: Multiple configs Add all files to $KUBECONFIG. You can do so in your shell config by adding

```
export KUBECONFIG=$KUBECONFIG:$HOME/.kube/config2
```

4.5.2 Access loadbalancers outside of the web UI

This uses the keystoneauth1 package for authentication and making API requests.

```
pip install keystoneauth1
```

```
import keystoneauth1
# Fill in your credentials
my_username = ""
my_password = ""
my_project_id = ""
```

(continues on next page)

(continued from previous page)

```
# Setting up a session
password_method = keystoneauth1.identity.v3.PasswordMethod(username=my_username,
password=my_password,
user_domain_name="GWDG")
auth = keystoneauth1.identity.v3.Auth(auth_url="https://api.prod.cloud.gwdg.de:5000/v3",␣
↪auth_methods=[password_method], project_id=my_project_id, project_domain_name="GWDG")
sess = keystoneauth1.session.Session(auth=auth)
# API docs: https://wiki.openstack.org/wiki/Neutron/LBaaS/API_2.0
lbaasv2_api = "https://api.prod.cloud.gwdg.de:9696/v2.0"
# List all Load Balancers
r = sess.request(f"{lbaasv2_api}/lbaas/loadbalancers", "GET",
 headers={"Accept": "application/json"})
print(r.text)
```

CHAPTER

FIVE

CREDITS

5.1 Development Lead

- Jonathan Decker <jonathan.decker@uni-goettingen.de>

5.2 Contributors

None yet. Why not be the first?

CHAPTER

SIX

CONTRIBUTOR COVENANT CODE OF CONDUCT

6.1 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, gender identity and expression, level of experience, nationality, personal appearance, race, religion, or sexual identity and orientation.

6.2 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

6.3 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

6.4 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

6.5 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by opening an issue. The project team will review and investigate all complaints, and will respond in a way that it deems appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

6.6 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at https://www.contributor-covenant.org/version/1/4/code-of-conduct.html

CHAPTER

SEVEN

IRONIK

7.1 ironik package

7.1.1 Subpackages

ironik.cli package

Submodules

ironik.cli.cli_helper module

author Jonathan Decker

ironik.cli.cli_helper.**check_and_prepare_cluster_name**(*name*)

Parameters **name** (*str*) –

Return type str

ironik.cli.cli_helper.**ensure_rancher_user**(*rancher_config*, *rancher_session*)

Parameters

- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **rancher_session** (*requests.sessions.Session*) –

Return type str

ironik.cli.cli_helper.**generate_random_string**(*length*)

Generates a secure random string of the given length and returns it. :param length: Length of the random string to return, only positive integers are allowed. :type length: int :return: A random string generated with the secrets built-in library. :rtype: str

Parameters **length** (*int*) –

Return type str

ironik.cli.cli_helper.**get_router_id_from_routers**(*public_network_id*, *routers*)

Parameters

- **public_network_id** (*str*) –

- **routers** (*list[dict]*) –

Returns

Return type str

ironik.cli.cli_helper.**handle_kubernetes_setup**(*openstack_credentials*, *openstack_config*, *rancher_config*, *rancher_session*, *cluster_id*, *subnet_id*, *public_network_id*, *router_id*)

Parameters

- **openstack_credentials** (ironik.config_file_handler.deploy_template.OpenStackCredentials) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **rancher_session** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **subnet_id** (*str*) –
- **public_network_id** (*str*) –
- **router_id** (*str*) –

Return type bool

ironik.cli.cli_helper.**remove_all_but_alphanum_dash_from_string_and_lower**(*dirty_string*)

Removes non-alphanumeric characters from the given string except dash "-" and lowers it before returning it. This is done to make user input comply with restrictions on names in the API. :param dirty_string: A string. :type dirty_string: str :return: The string without any non alphanumeric characters except dashes and in lower case. :rtype: str

Parameters **dirty_string** (*str*) –

Return type str

ironik.cli.cli_helper.**update_rancher_user**(*rancher_config*, *rancher_session*, *cluster_id*)

Parameters

- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **rancher_session** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –

Return type bool

ironik.cli.cli_helper.**validate_key_in_dict**(*key*, *list_dicts*)

Parameters

- **key** (*str*) –
- **list_dicts** (*list[dict]*) –

Returns

Return type bool

ironik.cli.cli_helper.**wait_for_cluster_ready**(*rancher_config*, *rancher_session*, *cluster_id*, *timeout_in_s=3600*)

Parameters

- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **rancher_session** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **timeout_in_s** (*int*) –

Return type bool

ironik.cli.cli_helper.**wait_for_nodes_ready**(*kubernetes_config*, *rancher_config*, *rancher_session*, *cluster_id*, *timeout_in_s=3600*)

Parameters

- **kubernetes_config** (ironik.config_file_handler.deploy_template.KubernetesConfig) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **rancher_session** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **timeout_in_s** (*int*) –

Return type bool

ironik.cli.ironik_cli module

Command-line interface.

author Jonathan Decker

ironik.cli.ironik_cli.**print_version**(*context*, *param*, *value*)

Parameters

- **context** –
- **param** –
- **value** –

Returns

Module contents

ironik.config_file_handler package

Submodules

ironik.config_file_handler.cloud_conf_parser module

author Jonathan Decker

ironik.config_file_handler.cloud_conf_parser.**config_ini_to_string**(*cloud_conf*)

Parameters **cloud_conf** (*configparser.ConfigParser*) –

Return type str

ironik.config_file_handler.cloud_conf_parser.**get_cloud_conf**(*openstack_credentials*, *openstack_config*, *subnet_id*, *public_network_id*, *router_id*)

Parameters

- **openstack_credentials** (ironik.config_file_handler.deploy_template.OpenStackCredentials) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –
- **subnet_id** (*str*) –
- **public_network_id** (*str*) –
- **router_id** (*str*) –

Returns

Return type configparser.ConfigParser

ironik.config_file_handler.deploy_template module

Definition and functions for spawning deployment templates.

author Jonathan Decker

class ironik.config_file_handler.deploy_template.**DeployConfig**(*rancher_credentials: ironik.config_file_handler.deploy_template.RancherCr = RancherCreden-tials(rancher_access_key='', rancher_secret_key=''), openstack_credentials: ironik.config_file_handler.deploy_template.OpenStack = OpenStackCreden-tials(username='', password='', project_id=''), rancher_config: ironik.config_file_handler.deploy_template.RancherCo = RancherCon-fig(rancher_api_base_url='', ssh_user='', new_cluster_admin_user_name='', new_cluster_admin_user_password='', engine_install_url='https://releases.rancher.com/insta docker/20.10.sh'), openstack_config: ironik.config_file_handler.deploy_template.OpenStack = OpenStackCon-fig(openstack_auth_url='', user_domain_name='', project_domain_name='', lb_provider='', default_flavor_name='', default_image_name='', remote_ip_prefix='', private_network_id='', region_name='', use_octavia=True, security_group_name='ironik-k8s-node', volume_size=10, volume_type='ssd'), network_config: ironik.config_file_handler.deploy_template.NetworkC = NetworkCon-fig(required_TCP_ports=[22, 80, 443, 2376, 2379, 2380, 6443, 8443, 8472, 9913, 10250, 10254], required_UDP_ports=[8443, 8472], worker_port_range_min=30000, worker_port_range_max=32767), kubernetes_config: ironik.config_file_handler.deploy_template.Kubernete = KubernetesCon-fig(master_node_roles='master, etcd, worker', worker_node_roles='worker', version='v1.22.9-rancher1-1', number_master_nodes=1, number_worker_nodes=1), deployment_options: ironik.config_file_handler.deploy_template.Deployme = DeploymentOp-tions(deploy_example_workload=False, example_workload_image='rancher/hello-world', example_workload_name='hello-world-example',*

Parameters

- **rancher_credentials** (ironik.config_file_handler.deploy_template.RancherCredentials) –
- **openstack_credentials** (ironik.config_file_handler.deploy_template.OpenStackCredentials) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –
- **network_config** (ironik.config_file_handler.deploy_template.NetworkConfig) –
- **kubernetes_config** (ironik.config_file_handler.deploy_template.KubernetesConfig) –
- **deployment_options** (ironik.config_file_handler.deploy_template.DeploymentOptions) –

Return type None

deployment_options: *ironik.config_file_handler.deploy_template.DeploymentOptions* = DeploymentOptions(deploy_example_workload=False, example_workload_image='rancher/hello-world', example_workload_name='hello-world-example', deploy_nginx_workload=False, nginx_ingress_version='4.1.0', nginx_ingress_repo='https://kubernetes.github.io/ingress-nginx', nginx_ingress_app_name='nginx-ingress-lb', install_cinder_driver=False, deploy_example_volume=False, cleanup_example_workload=True, cleanup_nginx_ingress=True, cleanup_example_volume=True)

kubernetes_config: *ironik.config_file_handler.deploy_template.KubernetesConfig* = KubernetesConfig(master_node_roles='master, etcd, worker', worker_node_roles='worker', version='v1.22.9-rancher1-1', number_master_nodes=1, number_worker_nodes=1)

network_config: *ironik.config_file_handler.deploy_template.NetworkConfig* = NetworkConfig(required_TCP_ports=[22, 80, 443, 2376, 2379, 2380, 6443, 8443, 8472, 9913, 10250, 10254], required_UDP_ports=[8443, 8472], worker_port_range_min=30000, worker_port_range_max=32767)

openstack_config: *ironik.config_file_handler.deploy_template.OpenStackConfig* = OpenStackConfig(openstack_auth_url='', user_domain_name='', project_domain_name='', lb_provider='', default_flavor_name='', default_image_name='', remote_ip_prefix='', private_network_id='', region_name='', use_octavia=True, security_group_name='ironik-k8s-node', volume_size=10, volume_type='ssd')

openstack_credentials: *ironik.config_file_handler.deploy_template.OpenStackCredentials* = OpenStackCredentials(username='', password='', project_id='')

rancher_config: *ironik.config_file_handler.deploy_template.RancherConfig* = RancherConfig(rancher_api_base_url='', ssh_user='', new_cluster_admin_user_name='', new_cluster_admin_user_password='', engine_install_url='https://releases.rancher.com/install-docker/20.10.sh')

rancher_credentials: *ironik.config_file_handler.deploy_template.RancherCredentials* = RancherCredentials(rancher_access_key='', rancher_secret_key='')

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!DeployConfig'

class ironik.config_file_handler.deploy_template.**DeploymentOptions**(*deploy_example_workload: bool = False*, *example_workload_image: str = 'rancher/hello-world'*, *example_workload_name: str = 'hello-world-example'*, *deploy_nginx_workload: bool = False*, *nginx_ingress_version: str = '4.1.0'*, *nginx_ingress_repo: str = 'https://kubernetes.github.io/ingress-nginx'*, *nginx_ingress_app_name: str = 'nginx-ingress-lb'*, *install_cinder_driver: bool = False*, *deploy_example_volume: bool = False*, *cleanup_example_workload: bool = True*, *cleanup_nginx_ingress: bool = True*, *cleanup_example_volume: bool = True*)

Bases: yaml.YAMLObject

Parameters

- **deploy_example_workload** (*bool*) –
- **example_workload_image** (*str*) –
- **example_workload_name** (*str*) –
- **deploy_nginx_workload** (*bool*) –
- **nginx_ingress_version** (*str*) –
- **nginx_ingress_repo** (*str*) –
- **nginx_ingress_app_name** (*str*) –
- **install_cinder_driver** (*bool*) –
- **deploy_example_volume** (*bool*) –
- **cleanup_example_workload** (*bool*) –
- **cleanup_nginx_ingress** (*bool*) –
- **cleanup_example_volume** (*bool*) –

Return type None

cleanup_example_volume: bool = True

cleanup_example_workload: bool = True

cleanup_nginx_ingress: bool = True

deploy_example_volume: bool = False

deploy_example_workload: bool = False

deploy_nginx_workload: bool = False

example_workload_image: str = 'rancher/hello-world'

example_workload_name: str = 'hello-world-example'

install_cinder_driver: bool = False

nginx_ingress_app_name: str = 'nginx-ingress-lb'

nginx_ingress_repo: str = 'https://kubernetes.github.io/ingress-nginx'

nginx_ingress_version: str = '4.1.0'

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!DeploymentOptions'

class ironik.config_file_handler.deploy_template.**KubernetesConfig**(*master_node_roles: str = 'master, etcd, worker'*, *worker_node_roles: str = 'worker'*, *version: str = 'v1.22.9-rancher1-1'*, *number_master_nodes: int = 1*, *number_worker_nodes: int = 1*)

Bases: yaml.YAMLObject

Parameters

- **master_node_roles** (*str*) –
- **worker_node_roles** (*str*) –
- **version** (*str*) –
- **number_master_nodes** (*int*) –
- **number_worker_nodes** (*int*) –

Return type None

master_node_roles: str = 'master, etcd, worker'

number_master_nodes: int = 1

number_worker_nodes: int = 1

version: str = 'v1.22.9-rancher1-1'

worker_node_roles: str = 'worker'

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!KubernetesConfig'

class ironik.config_file_handler.deploy_template.**NetworkConfig**(*required_TCP_ports: list = <factory>*, *required_UDP_ports: list = <factory>*, *worker_port_range_min: int = 30000*, *worker_port_range_max: int = 32767*)

Bases: yaml.YAMLObject

Parameters

- **required_TCP_ports** (*list[int]*) –
- **required_UDP_ports** (*list[int]*) –
- **worker_port_range_min** (*int*) –
- **worker_port_range_max** (*int*) –

Return type None

required_TCP_ports: list[int]

required_UDP_ports: list[int]

worker_port_range_max: int = 32767

worker_port_range_min: int = 30000

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!NetworkConfig'

class ironik.config_file_handler.deploy_template.**OpenStackConfig**(*openstack_auth_url: str*, *user_domain_name: str*, *project_domain_name: str*, *lb_provider: str*, *default_flavor_name: str*, *default_image_name: str*, *remote_ip_prefix: str*, *private_network_id: str*, *region_name: str = 'RegionOne'*, *use_octavia: bool = True*, *security_group_name: str = 'ironik-k8s-node'*, *volume_size: int = 10*, *volume_type: str = 'ssd'*)

Bases: yaml.YAMLObject

Parameters

- **openstack_auth_url** (*str*) –
- **user_domain_name** (*str*) –
- **project_domain_name** (*str*) –
- **lb_provider** (*str*) –
- **default_flavor_name** (*str*) –
- **default_image_name** (*str*) –
- **remote_ip_prefix** (*str*) –

- **private_network_id** (*str*) –
- **region_name** (*str*) –
- **use_octavia** (*bool*) –
- **security_group_name** (*str*) –
- **volume_size** (*int*) –
- **volume_type** (*str*) –

Return type None

default_flavor_name: str

default_image_name: str

lb_provider: str

openstack_auth_url: str

private_network_id: str

project_domain_name: str

region_name: str = 'RegionOne'

remote_ip_prefix: str

security_group_name: str = 'ironik-k8s-node'

use_octavia: bool = True

user_domain_name: str

volume_size: int = 10

volume_type: str = 'ssd'

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!OpenStackConfig'

class ironik.config_file_handler.deploy_template.**OpenStackCredentials**(*username: str*, *password: str*, *project_id: str*)

Bases: yaml.YAMLObject

Parameters

- **username** (*str*) –
- **password** (*str*) –
- **project_id** (*str*) –

Return type None

password: str

project_id: str

username: str

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!OpenStackCredentials'

class ironik.config_file_handler.deploy_template.**RancherConfig**(*rancher_api_base_url: str*, *ssh_user: str*, *new_cluster_admin_user_name: str*, *new_cluster_admin_user_password: str = ''*, *engine_install_url: str = 'https://releases.rancher.com/install-docker/20.10.sh'*)

Bases: yaml.YAMLObject

Parameters

- **rancher_api_base_url** (*str*) –
- **ssh_user** (*str*) –
- **new_cluster_admin_user_name** (*str*) –
- **new_cluster_admin_user_password** (*str*) –
- **engine_install_url** (*str*) –

Return type None

engine_install_url: str = 'https://releases.rancher.com/install-docker/20.10.sh'

new_cluster_admin_user_name: str

new_cluster_admin_user_password: str = ''

rancher_api_base_url: str

ssh_user: str

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!RancherConfig'

class ironik.config_file_handler.deploy_template.**RancherCredentials**(*rancher_access_key: str*, *rancher_secret_key: str*)

Bases: yaml.YAMLObject

Parameters

- **rancher_access_key** (*str*) –
- **rancher_secret_key** (*str*) –

Return type None

rancher_access_key: str

rancher_secret_key: str

yaml_loader
alias of yaml.loader.SafeLoader

yaml_tag = '!RancherCredentials'

ironik.config_file_handler.deploy_template.**load_deploy_template**(*path*)

Parameters **path** (*pathlib.Path*) –

Returns

Return type *ironik.config_file_handler.deploy_template.DeployConfig*

ironik.config_file_handler.deploy_template.**validate_deploy_template**()

ironik.config_file_handler.deploy_template.**write_deploy_template**(*path*, *overwrite*)

Parameters

- **path** (*pathlib.Path*) –
- **overwrite** (*bool*) –

Returns

Return type None

ironik.config_file_handler.manifest_parser module

author Jonathan Decker

ironik.config_file_handler.manifest_parser.**get_cloud_controller_role_bindings_manifest**()

Return type dict

ironik.config_file_handler.manifest_parser.**get_cloud_controller_roles_manifest**()

Return type dict

ironik.config_file_handler.manifest_parser.**get_csi_plugin_manifest**()

Return type list[dict]

ironik.config_file_handler.manifest_parser.**get_manifest_path**(*manifest*)

Parameters **manifest** (*str*) –

Return type pathlib.Path

ironik.config_file_handler.manifest_parser.**get_openstack_controller_manager_manifest**(*openstack_config*)

Parameters **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –

Return type list[dict]

Module contents

ironik.openstack_handler package

Submodules

ironik.openstack_handler.openstack_api_caller module

author Jonathan Decker

ironik.openstack_handler.openstack_api_caller.**associate_openstack_floating_ip_with_port_id**(*conn*, *public_network_id*, *target_port*)

Makes API calls to openstack to delete unattached floating IPs and create a new one and attach it to the given port. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :param public_network_id: Id of the public network from which the floating IP should come. :type public_network_id: str :param target_port: Id of the port, the new floating IP should be attached to. :type target_port: str :return: Dictionary or munch representing the newly assigned floating IP. :rtype: dict

Parameters

- **conn** (*openstack.connection.Connection*) –
- **public_network_id** (*str*) –
- **target_port** (*str*) –

Return type dict

ironik.openstack_handler.openstack_api_caller.**create_and_test_openstack_connection**(*openstack_credentials*, *openstack_config*)

Call create openstack connection and communicates possible errors. :param openstack_credentials: :param openstack_config:

Parameters

- **openstack_credentials** (ironik.config_file_handler.deploy_template.OpenStackCredentials) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –

Return type openstack.connection.Connection

ironik.openstack_handler.openstack_api_caller.**create_openstack_connection**(*username*, *password*, *project_id*, *auth_url*, *region_name*, *user_domain_name*, *project_domain_name*)

Creates an openstack.connection.Connection object based on the given credentials and further information from gwdg_defaults. This alone does not validate any of the credentials. This is the base object for all API calls to Openstack using the openstacksdk. Docs can be found here: https://docs.openstack.org/openstacksdk/latest/user/connection.html :param username: Openstack username. :type username: str :param password: Openstack password. :type password: str :param project_id: Openstack project id for which this tool should run. :type project_id: str :param auth_url: Openstack authentication url, which should be the identity service url followed by /v3. :type auth_url: str :return: A connection object from the openstacksdk. :rtype: openstack.connection.Connection :param region_name: :param user_domain_name: :param project_domain_name:

Parameters

- **username** (*str*) –
- **password** (*str*) –
- **project_id** (*str*) –

- **auth_url** (*str*) –
- **region_name** (*str*) –
- **user_domain_name** (*str*) –
- **project_domain_name** (*str*) –

Return type openstack.connection.Connection

ironik.openstack_handler.openstack_api_caller.**create_openstack_security_group**(*conn*, *openstack_config*, *network_config*, *cluster_name*)

Checks whether a security group with the given name already exists and if yes tries to append the cluster name to it. If that security group also exists, deletes it and recreates it. Finally, returns the name of the newly created security group. :param conn: Valid openstack connection :type conn: openstack.connection.Connection :param openstack_config: :param network_config: :param cluster_name: :return: :rtype: bool

Parameters

- **conn** (*openstack.connection.Connection*) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –
- **network_config** (ironik.config_file_handler.deploy_template.NetworkConfig) –
- **cluster_name** (*str*) –

Return type bool

ironik.openstack_handler.openstack_api_caller.**find_floating_ip_for_internal_ip**(*conn*, *internal_ip*)

Parameters

- **conn** (*openstack.connection.Connection*) – A connection object initialized by create_openstack_connection.
- **internal_ip** (*str*) – The internal IP inside OpenStack of an instance or load balancer.

Returns The respective floating IP associated with the internal IP or an empty string.

Return type str

ironik.openstack_handler.openstack_api_caller.**find_openstack_load_balancer_port_id_by_ip**(*conn*, *openstack_config*, *load_balancer_int*

Makes a call to the openstack API to find the id of port belonging to a load balancer with the given ip. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :param openstack_config: :param load_balancer_internal_ip: Ip of the load balancer device. :type load_balancer_internal_ip: str :return: Id of the target port. :rtype: str

Parameters

- **conn** (*openstack.connection.Connection*) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –

- **load_balancer_internal_ip** (*str*) –

Return type str

ironik.openstack_handler.openstack_api_caller.**find_out_openstack_floating_ip_is_free**(*conn*)
Make a call to the openstack API to get the current floating IPs and check whether an IP is not attached. It further tries to claim another floating IP to check whether quota is still left for additional floating IPs in case all listed floating IPs are already attached. Returns a tuple with the number of attached IPs and a bool than indicates whether at least one floating IP is free. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A tuple with an int that represents the number of attached IPs and a bool that indicates whether at least one floating IP is free :rtype: (int, bool)

Parameters **conn** (*openstack.connection.Connection*) –

Return type Tuple[int, bool]

ironik.openstack_handler.openstack_api_caller.**get_openstack_compute_limits**(*conn*)
Makes a call to the openstack API for the compute limits, converts it into a dictionary and returns it. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A dictionary containing the compute limits of the openstack account. :rtype: dict

Parameters **conn** (*openstack.connection.Connection*) –

Return type dict

ironik.openstack_handler.openstack_api_caller.**get_openstack_flavors**(*conn*)
Makes a call to the openstack API to receive the available flavors and filters the returned attributes. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A list of dictionaries each describing a flavor. :rtype: list[dict]

Parameters **conn** (*openstack.connection.Connection*) –

Return type list[dict]

ironik.openstack_handler.openstack_api_caller.**get_openstack_images**(*conn*)
Makes a call to the openstack API to receive the available images and filter the returned attributes. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A list of dictionaries each describing an image. :rtype: list[dict]

Parameters **conn** (*openstack.connection.Connection*) –

Return type list[dict]

ironik.openstack_handler.openstack_api_caller.**get_openstack_public_and_private_networks**(*conn*)
Makes a request to the openstack API for all networks and finds the public network and declares all other networks as private networks. Returns a tuple with the public network and the list of private networks. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A tuple with a dictionary for the public network and a list of dictionaries for the private networks. :rtype: (dict, list[dict])

Parameters **conn** (*openstack.connection.Connection*) –

Return type Tuple[dict, list[dict]]

ironik.openstack_handler.openstack_api_caller.**get_openstack_routers**(*conn*)
Make a request to the Openstack API to get all routers, filter some attributes and return a list of dictionaries. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A list of dictionaries where each entry represents a router. :rtype: list[dict]

Parameters **conn** (*openstack.connection.Connection*) –

Return type list[dict]

ironik.openstack_handler.openstack_api_caller.**get_openstack_subnets**(*conn*)
Makes a request to the openstack API to get all subnets, filter some attributes and return a list of dictionaries. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: A list of dictionaries where each entry represents a subnet. :rtype: list[dict]

Parameters **conn** (*openstack.connection.Connection*) –

Return type list[dict]

ironik.openstack_handler.openstack_api_caller.**get_openstack_volume_limits**(*conn*)
Returns a dict with the volume limits for the given OpenStack connection.

Parameters

- **(conn** (*conn*) – openstack.connection.Connection): A connection object initialized by create_openstack_connection.
- **conn** (*openstack.connection.Connection*) –

Returns A dictionary containing the volume limits of the openstack account.

Return type dict

ironik.openstack_handler.openstack_api_caller.**match_subnet**(*conn*, *network_id*)
Calls get openstack subnets and matches the subnet with the given network id or calls ask for subnet if there are multiple subnets. :param conn: Valid openstack connection :type conn: openstack.connection.Connection :param network_id: Network Id of the external network of the GWDG openstack deployment. :type network_id: str :return: Dictionary representing the subnet :rtype: dict

Parameters

- **conn** (*openstack.connection.Connection*) –
- **network_id** (*str*) –

Return type dict

ironik.openstack_handler.openstack_api_caller.**verify_openstack_connection**(*conn*)
Verifies that the openstack connection is valid by making a simple API call. Returns True if it is valid and false otherwise. :param conn: A connection object initialized by create_openstack_connection. :type conn: openstack.connection.Connection :return: True if the connection is valid and false otherwise. :rtype: bool

Parameters **conn** (*openstack.connection.Connection*) –

Return type bool

ironik.openstack_handler.resource_calculator module

author Jonathan Decker

ironik.openstack_handler.resource_calculator.**calculate_free_resources**(*conn*, *flavor_dict*, *kubernetes_config*, *openstack_config*, *check_ip*)
Check whether compute limits can satisfy requested deployment and give an overview.

Calls get openstack compute limits, as well as get openstack volume limits and calculates how many resources will be used by deploying the given number of master and worker nodes with the given flavors and volumes. Also displays an overview.

Parameters

- **conn** (*openstack.connection.Connection*) – Valid openstack connection.

- **flavor_dict** (*dict*) – Dictionary of the master flavor as returned by ask for flavor.
- **kubernetes_config** (KubernetesConfig) –
- **openstack_config** (OpenStackConfig) –
- **check_ip** (*bool*) – Whether to skip checking for free floating IPs, will skip if False.

Returns A bool that is true if the required resources fit within the maximum allowed compute limits

Return type bool

ironik.openstack_handler.resource_calculator.**preprocess_and_calculate_resource_consumption**(*openstack_config*, *deployment_options*, *kubernetes_config*, *openstack_connection*, *flavors*, *images*)

Parameters

- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –
- **deployment_options** (ironik.config_file_handler.deploy_template.DeploymentOptions) –
- **kubernetes_config** (ironik.config_file_handler.deploy_template.KubernetesConfig) –
- **openstack_connection** (*openstack.connection.Connection*) –
- **flavors** (*list[dict]*) –
- **images** (*list[dict]*) –

Return type bool

Module contents

ironik.rancher package

Submodules

ironik.rancher.kubernetes_api_caller module

author Jonathan Decker

ironik.rancher.kubernetes_api_caller.**apply_controller_manager_manifests**(*kube_client*, *openstack_config*)

Parameters

- **kube_client** (*kubernetes.client.api.core_v1_api.CoreV1Api*) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –

Return type bool

ironik.rancher.kubernetes_api_caller.**apply_csi_driver_manifests**(*kube_client*)

Parameters **kube_client** (*kubernetes.client.api.core_v1_api.CoreV1Api*) –

Return type bool

ironik.rancher.kubernetes_api_caller.**create_cloud_conf_secret**(*kube_client*, *cloud_conf_str*)

Parameters

- **kube_client** (*kubernetes.client.api.core_v1_api.CoreV1Api*) –
- **cloud_conf_str** (*str*) –

Return type bool

ironik.rancher.kubernetes_api_caller.**init_client**(*kube_config*)

Parameters **kube_config** (*str*) –

Returns

Return type kubernetes.client.api.core_v1_api.CoreV1Api

ironik.rancher.kubernetes_api_caller.**verify_client**(*kube_client*)

Parameters **kube_client** (*kubernetes.client.api.core_v1_api.CoreV1Api*) –

Return type bool

ironik.rancher.rancher_api_caller module

author Jonathan Decker

ironik.rancher.rancher_api_caller.**add_rancher_base_binding_to_user**(*s*, *rancher_config*, *user_id*)
Makes an API call to rancher to add the user-base role binding to the given user id. :param rancher_config: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param user_id: :return: True if it worked and false otherwise. :rtype: bool

Parameters

- **s** (*requests.sessions.Session*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **user_id** (*str*) –

Return type bool

ironik.rancher.rancher_api_caller.**check_rancher_cluster_ready**(*s*, *cluster_id*, *rancher_config*)

Parameters

- **s** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Returns

Return type bool

ironik.rancher.rancher_api_caller.**check_rancher_nodes_ready**(*s*, *cluster_id*, *rancher_config*)

Checks how many nodes for the given cluster are in an active state and returns the number. :param rancher_config: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param cluster_id: Id of the cluster to which the nodes should belong. :type cluster_id: str :return: The number of active nodes. :rtype: init

Parameters

- **s** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Return type int

ironik.rancher.rancher_api_caller.**create_and_test_rancher_session**(*rancher_credentials*, *rancher_config*)

Calls create rancher session and communicates possible errors. :param rancher_config: :param rancher_credentials: :return: Returns a requests Session which is configured to authenticate with rancher or None if the session is not valid. :rtype: requests.Session or None

Parameters

- **rancher_credentials** (ironik.config_file_handler.deploy_template.RancherCredentials) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Return type requests.sessions.Session

ironik.rancher.rancher_api_caller.**create_rancher_cluster**(*s*, *rancher_config*, *cluster_name*, *iso_time_stamp*, *rke_config*)

Makes an API call to rancher to create a new cluster using the given parameters. :param rancher_config: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param cluster_name: The name for the cluster in Rancher. :type cluster_name: str :param iso_time_stamp: Iso time stamp which is written to the description of the new cluster. :type iso_time_stamp: str :param rke_config: Rancher kubernetes engine configuration as produced by rke template. :type rke_config: dict :return: The id of the new cluster in rancher. :rtype: str

Parameters

- **s** (*requests.sessions.Session*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **cluster_name** (*str*) –
- **iso_time_stamp** (*str*) –

- **rke_config** (*dict*) –

Return type str

ironik.rancher.rancher_api_caller.**create_rancher_node_pool**(*s*, *cluster_name*, *cluster_id*, *node_template_id*, *is_master*, *kubernetes_config*, *rancher_config*)

Makes an API call the rancher API to create a new node pool for the given cluster with the given node template. Quantity of 0 will be accepted but the new node pool will not show up in the UI. :param rancher_config: :param kubernetes_config: :param cluster_name: Name of the cluster, used to prefix the name of the nodes. :type cluster_name: str :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param cluster_id: Id of the cluster in rancher. :type cluster_id: str :param node_template_id: Id of the node template to use for the cluster. :type node_template_id: str :param is_master: Indicates whether the nodes are supposed to be master nodes for the kubernetes cluster or just worker nodes. Master nodes also get the roles etcd and controlPlane in addition to worker. Worker only get the worker role. Further the name prefix for master nodes is master and worker for worker. :type is_master: bool :return: Id of the node pool in rancher. :rtype: str

Parameters

- **s** (*requests.sessions.Session*) –
- **cluster_name** (*str*) –
- **cluster_id** (*str*) –
- **node_template_id** (*str*) –
- **is_master** (*bool*) –
- **kubernetes_config** (ironik.config_file_handler.deploy_template.KubernetesConfig) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Return type str

ironik.rancher.rancher_api_caller.**create_rancher_node_template**(*s*, *openstack_credentials*, *openstack_config*, *rancher_config*, *template_name*)

Queries rancher API to create a new node template based on the given instructions and returns the id of the template. :param rancher_config: :param openstack_config: :param openstack_credentials: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param template_name: The name of the template in rancher. :type template_name: str :return: Id of the node template created in rancher. :rtype: str

Parameters

- **s** (*requests.sessions.Session*) –
- **openstack_credentials** (ironik.config_file_handler.deploy_template.OpenStackCredentials) –
- **openstack_config** (ironik.config_file_handler.deploy_template.OpenStackConfig) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **template_name** (*str*) –

Return type str

ironik.rancher.rancher_api_caller.**create_rancher_session**(*rancher_access_key*, *rancher_secret_key*, *disable_verify=True*)

Creates a requests Session used to access Rancher API. This needs to be forwarded to any function making API calls to Rancher. SSL verification is disabled by default as the test server does not have a valid ssl certificate. :param rancher_access_key: :param rancher_secret_key: :param disable_verify: Determines whether ssl verification and warnings should be disabled. :type disable_verify: bool :return: A Session object used to make API calls to Rancher. :rtype: requests.Session

Parameters

- **rancher_access_key** (*str*) –
- **rancher_secret_key** (*str*) –
- **disable_verify** (*bool*) –

Return type requests.sessions.Session

ironik.rancher.rancher_api_caller.**create_rancher_user**(*s*, *rancher_config*)

Makes an API call to rancher to create a new user in rancher with the given name and password, setting must change password on first login to true. :param rancher_config: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :return: Id of the user. :rtype: str

Parameters

- **s** (*requests.sessions.Session*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Return type str

ironik.rancher.rancher_api_caller.**get_rancher_kubernetes_engine_config**(*kubernetes_config*, *network_config*)

Parameters

- **kubernetes_config** (ironik.config_file_handler.deploy_template.KubernetesConfig) –
- **network_config** (ironik.config_file_handler.deploy_template.NetworkConfig) –

Return type dict

ironik.rancher.rancher_api_caller.**get_rancher_user_id**(*s*, *rancher_config*)

Queries the rancher API to check if a user exists and returns true if it does and false otherwise. :param rancher_config: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :return: :rtype:

Parameters

- **s** (*requests.sessions.Session*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Return type str

ironik.rancher.rancher_api_caller.**get_rke_template**(*kubernetes_config*, *network_config*)

Parameters

- **kubernetes_config** (ironik.config_file_handler.deploy_template.KubernetesConfig) –
- **network_config** (ironik.config_file_handler.deploy_template.NetworkConfig) –

Returns

Return type dict

ironik.rancher.rancher_api_caller.**make_rancher_user_cluster_owner**(*s*, *cluster_id*, *rancher_config*, *user_id*)

Makes an API call to rancher to make the given user id owner of the given cluster. :param rancher_config: :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param cluster_id: Id of the cluster. :type cluster_id: str :param user_id: :return: True if it worked and false otherwise. :rtype: bool

Parameters

- **s** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –
- **user_id** (*str*) –

Return type bool

ironik.rancher.rancher_api_caller.**request_kubeconfig**(*s*, *cluster_id*, *rancher_config*)

Parameters

- **s** (*requests.sessions.Session*) –
- **cluster_id** (*str*) –
- **rancher_config** (ironik.config_file_handler.deploy_template.RancherConfig) –

Returns

Return type str

ironik.rancher.rancher_api_caller.**verify_rancher_token**(*s*, *base_url*)

Verifies whether a Rancher token is valid by trying to get the given base url using the given Session object. Returns True if everything is fine and False if either the token is invalid or the url is unreachable. :param s: A Session object initialized by create_rancher_session. :type s: requests.Session :param base_url: Base url of the Rancher API, should be the URL of the Rancher server with /v3 attached to it. :type base_url: str :return: True if the token is valid and false if either the token is invalid or the url is unreachable. :rtype: bool

Parameters

- **s** (*requests.sessions.Session*) –
- **base_url** (*str*) –

Return type bool

Module contents

ironik.util package

Submodules

ironik.util.exceptions module

author Jonathan Decker

exception ironik.util.exceptions.**IronikFatalError**(*message*)
Bases: Exception

exception ironik.util.exceptions.**IronikPassingError**(*message*)
Bases: Exception

ironik.util.exceptions.**passing_error_handler**(*func*)

ironik.util.ironik_logger module

Offers logging utilities for the entire program. Should be imported in every file as

"logger = logging.getLogger("logger")"

author Jonathan Decker

ironik.util.ironik_logger.**setup_logger**(*logger_name='logger'*, *console_level=50*, *log_file_level=20*, *logs_to_keep=20*, *create_log_files=True*, *path_to_logs=None*)

Description Sets up a logger with formatting, log file creation, settable console and log file logging levels as well as automatic deletion of old log files.

Parameters

- **logger_name** (*str*) – Name of the logger profile, standard is logger, this should not be changed except for testing
- **console_level** (*logging level*) – logging level on console, standard is INFO
- **log_file_level** (*logging level*) – logging level in log file, standard is DEBUG
- **logs_to_keep** (*int*) – number of log files to keep before deleting the oldest one, standard is 20
- **create_log_files** (*bool*) – Whether to create log files or only log to console
- **path_to_logs** (*pathlib.PATH*) – path to the directory for saving the logs, standard is current working directory / logs

Returns None, but the logger may now be accessed via logging.getLogger(logger_name), standard is logger

Return type None

7.1.2 Module contents

Top-level package for ironik.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

i

INDEX

F

G

H

I

K

L

M

N

O

P

R

S

U

V

W

Y