OpenStack-Ansible Documentation: os_ironic role

Release 18.1.0.dev197

OpenStack-Ansible Contributors

Dec 18, 2020

CONTENTS

CHAPTER

ONE

CONFIGURING THE BARE METAL (IRONIC) SERVICE (OPTIONAL)

Note: This feature is experimental at this time and it has not been fully production tested yet. These implementation instructions assume that ironic is being deployed as the sole hypervisor for the region.

Ironic is an OpenStack project which provisions bare metal (as opposed to virtual) machines by leveraging common technologies such as PXE boot and IPMI to cover a wide range of hardware, while supporting pluggable drivers to allow vendor-specific functionality to be added.

OpenStacks ironic project makes physical servers as easy to provision as virtual machines in a cloud.

1.1 OpenStack-Ansible deployment

1. Modify the environment files and force `nova-compute` to run from within a container:

```
sed -i '/is_metal.*/d' /etc/openstack_deploy/env.d/nova.yml
```

1.2 Setup a neutron network for use by ironic

In a general case, neutron networking can be a simple flat network. However, in a complex case, this can be whatever you need and want. Ensure you adjust the deployment accordingly. The following is an example:

```
neutron net-create cleaning-net --shared \
                                --provider:network_type flat \
                                --provider:physical_network ironic-net

neutron subnet-create ironic-net 172.19.0.0/22 --name ironic-subnet
                      --ip-version=4 \
                      --allocation-pool start=172.19.1.100,end=172.19.1.
↪200 \
                      --enable-dhcp \
                      --dns-nameservers list=true 8.8.4.4 8.8.8.8
```

1.3 Building ironic images

Images using the `diskimage-builder` must be built outside of a container. For this process, use one of the physical hosts within the environment.

1. Install the necessary packages:

```
apt-get install qemu uuid-runtime curl
```

2. Install the `disk-imagebuilder` package:

```
pip install diskimage-builder --isolated
```

Important: Only use the `--isolated` flag if you are building on a node deployed by OpenStack-Ansible, otherwise pip will not resolve the external package.

3. Optional: Force the ubuntu `image-create` process to use a modern kernel:

```
echo 'linux-image-generic-lts-xenial:' > \
/usr/local/share/diskimage-builder/elements/ubuntu/package-installs.
↪yaml
```

4. Create Ubuntu `initramfs`:

```
disk-image-create ironic-agent ubuntu -o ${IMAGE_NAME}
```

5. Upload the created deploy images into the Image (glance) Service:

```
# Upload the deploy image kernel
glance image-create --name ${IMAGE_NAME}.kernel --visibility public \
 --disk-format aki --container-format aki < ${IMAGE_NAME}.kernel

# Upload the user image initramfs
glance image-create --name ${IMAGE_NAME}.initramfs --visibility␣
↪public \
 --disk-format ari --container-format ari < ${IMAGE_NAME}.initramfs
```

6. Create Ubuntu user image:

```
disk-image-create ubuntu baremetal localboot local-config dhcp-all-
↪interfaces grub2 -o ${IMAGE_NAME}
```

7. Upload the created user images into the Image (glance) Service:

```
# Upload the user image vmlinuz and store uuid
VMLINUZ_UUID="$(glance image-create --name ${IMAGE_NAME}.vmlinuz --
↪visibility public --disk-format aki --container-format aki  < $
↪{IMAGE_NAME}.vmlinuz | awk '/\| id/ {print $4}')"

# Upload the user image initrd and store uuid
INITRD_UUID="$(glance image-create --name ${IMAGE_NAME}.initrd --
↪visibility public --disk-format ari --container-format ari  < $
↪{IMAGE_NAME}.initrd | awk '/\| id/ {print $4}')"
```

(continues on next page)

(continued from previous page)

```
# Create image
glance image-create --name ${IMAGE_NAME} --visibility public --disk-
↪format qcow2 --container-format bare --property kernel_id=${VMLINUZ_
↪UUID} --property ramdisk_id=${INITRD_UUID} < ${IMAGE_NAME}.qcow2
```

1.4 Creating an ironic flavor

1. Create a new flavor called `my-baremetal-flavor`.

Note: The following example sets the CPU architecture for the newly created flavor to be *x86_64*.

```
nova flavor-create ${FLAVOR_NAME} ${FLAVOR_ID} ${FLAVOR_RAM} ${FLAVOR_
↪DISK} ${FLAVOR_CPU}
nova flavor-key ${FLAVOR_NAME} set cpu_arch=x86_64
nova flavor-key ${FLAVOR_NAME} set capabilities:boot_option="local"
```

Note: Ensure the flavor and nodes match when enrolling into ironic. See the documentation on flavors for more information: https://docs.openstack.org/nova/queens/admin/flavors.html

After successfully deploying the ironic node on subsequent boots, the instance boots from your local disk as first preference. This speeds up the deployed nodes boot time. Alternatively, if this is not set, the ironic node PXE boots first and allows for operator-initiated image updates and other operations.

Note: The operational reasoning and building an environment to support this use case is not covered here.

1.4.1 Enroll ironic nodes

1. From the utility container, enroll a new baremetal node by executing the following:

```
# Source credentials
. ~/openrc

# Create the node
NODE_HOSTNAME="myfirstnodename"
IPMI_ADDRESS="10.1.2.3"
IPMI_USER="my-ipmi-user"
IPMI_PASSWORD="my-ipmi-password"
KERNEL_IMAGE=$(glance image-list | awk "/${IMAGE_NAME}.kernel/ {print␣
↪\$2}")
INITRAMFS_IMAGE=$(glance image-list | awk "/${IMAGE_NAME}.initramfs/
↪{print \$2}")
ironic node-create \
      -d agent_ipmitool \
      -i ipmi_address="${IPMI_ADDRESS}" \
```

(continues on next page)

(continued from previous page)

```
      -i ipmi_username="${IPMI_USER}" \
      -i ipmi_password="${IPMI_PASSWORD}" \
      -i deploy_ramdisk="${INITRAMFS_IMAGE}" \
      -i deploy_kernel="${KERNEL_IMAGE}" \
      -n ${NODE_HOSTNAME}

# Create a port for the node
NODE_MACADDRESS="aa:bb:cc:dd:ee:ff"
ironic port-create \
      -n $(ironic node-list | awk "/${NODE_HOSTNAME}/ {print \$2}") \
      -a ${NODE_MACADDRESS}

# Associate an image to the node
ROOT_DISK_SIZE_GB=40
ironic node-update $(ironic node-list | awk "/${IMAGE_NAME}/ {print \
↪$2}") add \
    driver_info/deploy_kernel=$KERNEL_IMAGE \
    driver_info/deploy_ramdisk=$INITRAMFS_IMAGE \
    instance_info/deploy_kernel=$KERNEL_IMAGE \
    instance_info/deploy_ramdisk=$INITRAMFS_IMAGE \
    instance_info/root_gb=${ROOT_DISK_SIZE_GB}

# Add node properties
# The property values used here should match the hardware used
ironic node-update $(ironic node-list | awk "/${NODE_HOSTNAME}/
↪{print \$2}") add \
    properties/cpus=48 \
    properties/memory_mb=254802 \
    properties/local_gb=80 \
    properties/size=3600 \
    properties/cpu_arch=x86_64 \
    properties/capabilities=memory_mb:254802,local_gb:80,cpu_arch:x86_
↪64,cpus:48,boot_option:local
```

1.5 Deploy a baremetal node kicked with ironic

Important: You will not have access unless you have a key set within nova before your ironic deployment. If you do not have an ssh key readily available, set one up with `ssh-keygen`.

```
nova keypair-add --pub-key ~/.ssh/id_rsa.pub admin
```

Now boot a node:

```
nova boot --flavor ${FLAVOR_NAME} --image ${IMAGE_NAME} --key-name admin $
↪{NODE_NAME}
```

CHAPTER

TWO

CONFIGURING THE BARE METAL (IRONIC) INSPECTOR SERVICE (OPTIONAL)

Note: This feature is experimental at this time and it has not been fully production tested yet.

Ironic Inspector is an Ironic service that deploys a tiny image called ironic-python-agent that gathers information about a Bare Metal node. The data is then stored in the database for further use later. The node is then updated with properties based in the introspection data.

The inspector configuration requires some pre-deployment steps to allow the Ironic playbook to make the inspector functioning.

2.1 Networking

Ironic networking must be configured as normally done. The inspector and Ironic will both share the TFTP server.

Networking will depend heavily on your environment. For example, the DHCP for both Ironic and inspector will come from the same subnet and will be a subset of the typical ironic allocated range.

2.2 Required Overrides

```
# names of your ironic-python-agent initrd/kernel images
ironic_inspector_ipa_initrd_name: ironic-deploy.initramfs
ironic_inspector_ipa_kernel_name: ironic-deploy.vmlinuz

# dnsmasq/dhcp information for inspector
ironic_inspector_dhcp_pool_range: <START> <END> (subset of ironic␣
↪IPs)
ironic_inspector_dhcp_subnet: <IRONIC SUBNET CIDR>
ironic_inspector_dhcp_subnet_mask: 255.255.252.0
ironic_inspector_dhcp_gateway: <IRONIC GATEWAY>
ironic_inspector_dhcp_nameservers: 8.8.8.8
```

This is an OpenStack-Ansible role to deploy the Bare Metal (ironic) service. See the role-ironic spec for more information.

To clone or view the source code for this repository, visit the role repository for os_ironic.

CHAPTER

THREE

DEFAULT VARIABLES

```
# Defaults file for openstack-ansible-ironic

# Verbosity Options
debug: False

#python venv executable
ironic_venv_python_executable: "{{ openstack_venv_python_executable |␣
↪default('python2') }}"

# Set the host which will execute the shade modules
# for the service setup. The host must already have
# clouds.yaml properly configured.
ironic_service_setup_host: "{{ openstack_service_setup_host | default(
↪'localhost') }}"
ironic_service_setup_host_python_interpreter: "{{ openstack_service_setup_
↪host_python_interpreter | default((ironic_service_setup_host ==
↪'localhost') | ternary(ansible_playbook_python, ansible_python[
↪'executable'])) }}"

# Comma separated list of Glance API servers
ironic_glance_api_servers: "{{ (glance_service_internalurl | default(
↪'http://localhost')) | urlsplit('scheme') ~ '://' ~ (glance_service_
↪internalurl | default('http://localhost')) | urlsplit('netloc') }}"

# Set the package install state for distribution and pip packages
# Options are 'present' and 'latest'
ironic_package_state: "latest"
ironic_pip_package_state: "latest"

ironic_git_repo: https://opendev.org/openstack/ironic
ironic_inspector_git_repo: https://opendev.org/openstack/ironic-inspector
ironic_git_install_branch: master
ironic_inspector_git_install_branch: master
ironic_upper_constraints_url: "{{ requirements_git_url | default('https://
↪releases.openstack.org/constraints/upper/' ~ requirements_git_install_
↪branch | default('master')) }}"
ironic_git_constraints:
  - "git+{{ ironic_git_repo }}@{{ ironic_git_install_branch }}#egg=ironic"
  - "git+{{ ironic_inspector_git_repo }}@{{ ironic_inspector_git_install_
↪branch }}#egg=ironic-inspector"
  - "--constraint {{ ironic_upper_constraints_url }}"

ironic_pip_install_args: "{{ pip_install_options | default('') }}"
```

(continues on next page)

(continued from previous page)

```
# Name of the virtual env to deploy into
ironic_venv_tag: "{{ venv_tag | default('untagged') }}"
ironic_bin: "/openstack/venvs/ironic-{{ ironic_venv_tag }}/bin"

# System info
ironic_system_user_name: ironic
ironic_system_group_name: ironic
ironic_system_shell: /bin/bash
ironic_system_comment: ironic system user
ironic_system_home_folder: "/var/lib/{{ ironic_system_user_name }}"
ironic_lock_path: /var/lock/ironic

# Ironic Program and Service names
python_ironic_client_program_name: ironic
ironic_services:
  ironic-api:
    group: ironic_api
    service_name: ironic-api
    init_config_overrides: "{{ ironic_api_init_config_overrides }}"
    wsgi_app: True
    wsgi_name: ironic-api-wsgi
    uwsgi_overrides: "{{ ironic_api_uwsgi_ini_overrides }}"
    uwsgi_port: "{{ ironic_service_port }}"
    uwsgi_bind_address: "{{ ironic_uwsgi_bind_address }}"
  ironic-conductor:
    group: ironic_conductor
    service_name: ironic-conductor
    init_config_overrides: "{{ ironic_conductor_init_config_overrides }}"
    execstarts: "{{ ironic_bin }}/ironic-conductor"
  ironic-inspector:
    group: ironic_inspector
    service_name: ironic-inspector
    init_config_overrides: "{{ ironic_inspector_init_config_overrides }}"
    execstarts: "{{ ironic_bin }}/ironic-inspector"

ironic_service_name: ironic
ironic_service_type: baremetal
ironic_service_proto: http
ironic_service_publicuri_proto: "{{ openstack_service_publicuri_proto |␣
↪default(ironic_service_proto) }}"
ironic_service_adminuri_proto: "{{ openstack_service_adminuri_proto |␣
↪default(ironic_service_proto) }}"
ironic_service_internaluri_proto: "{{ openstack_service_internaluri_proto␣
↪| default(ironic_service_proto) }}"
ironic_service_port: 6385
ironic_service_description: "Ironic baremetal provisioning service"
ironic_service_publicuri: "{{ ironic_service_publicuri_proto }}://{{␣
↪external_lb_vip_address }}:{{ ironic_service_port }}"
ironic_service_publicurl: "{{ ironic_service_publicuri }}"
ironic_service_adminuri: "{{ ironic_service_adminuri_proto }}://{{␣
↪internal_lb_vip_address }}:{{ ironic_service_port }}"
ironic_service_adminurl: "{{ ironic_service_adminuri }}"
ironic_service_internaluri: "{{ ironic_service_internaluri_proto }}://{{␣
↪internal_lb_vip_address }}:{{ ironic_service_port }}"
ironic_service_internalurl: "{{ ironic_service_internaluri }}"
```

(continues on next page)

(continued from previous page)

```
ironic_program_name: ironic-api
ironic_service_region: RegionOne
ironic_service_project_name: "service"
ironic_service_project_domain_id: default
ironic_service_user_domain_id: default
ironic_service_role_name: "admin"
ironic_service_in_ldap: False

# Ironic image store information
#
### Hosted Web Server
#
# Set this to True to use http web server to host floppy
# images and generated boot ISO. This requires http_root and
# http_url to be configured in the [deploy] section of the
# config file. If this is set to False, then Ironic will use
# Swift to host the floppy images and generated boot_iso.
ironic_enable_web_server_for_images: False
ironic_http_url: "{{ ironic_ipxe_proto }}://{{ ansible_host }}:{{ ironic_
↪ipxe_port }}"
ironic_http_root: "/httpboot"
#
### Swift Config
#
ironic_swift_image_container: glance_images
ironic_swift_api_version: v1
ironic_swift_url_endpoint_type: swift
# The ironic swift auth account and swift endpoints will be generated␣
↪using the
#  known swift data as provided by swift stat. If you wish to set either␣
↪of these
#  items to something else define these variables.
# ironic_swift_auth_account: AUTH_1234567890
# ironic_swift_endpoint: https://localhost:8080

# Is this Ironic installation working standalone?
# If you're wanting Ironic to work without being integrated to other␣
↪OpenStack
# services, set this to True, and update the dhcp configuration␣
↪appropriately
ironic_standalone: False

# Enables or disables automated cleaning. Automated cleaning
# is a configurable set of steps, such as erasing disk drives,
# that are performed on the node to ensure it is in a baseline
# state and ready to be deployed to.
ironic_automated_clean: false
# Set to 0 to disable erase devices on cleaning
ironic_erase_devices_priority: 10

# Database
ironic_db_setup_host: "{{ openstack_db_setup_host | default('localhost') }}
↪"
ironic_db_setup_python_interpreter: "{{ openstack_db_setup_python_
↪interpreter | default((ironic_db_setup_host == 'localhost') |␣
↪ternary(ansible_playbook_python, ansible_python['executable'])) }}"
```

(continues on next page)

(continued from previous page)

```
ironic_galera_address: "{{ galera_address | default('127.0.0.1') }}"
ironic_galera_user: ironic
ironic_galera_database: ironic
ironic_galera_use_ssl: "{{ galera_use_ssl | default(False) }}"
ironic_galera_ssl_ca_cert: "{{ galera_ssl_ca_cert | default('/etc/ssl/
↪certs/galera-ca.pem') }}"
ironic_galera_port: 3306

## Keystone authentication middleware
ironic_keystone_auth_plugin: password

# Neutron network - Set these in a playbook/task - can be set manually.
# Only "name" or "uuid" is needed, uuid will take preference if both are␣
↪specified.
# The cleaning and inspection network is not required to be set; they will␣
↪default
# to the provisioning network if not specified.
# ironic_neutron_provisioning_network_uuid: "UUID for provisioning network␣
↪in neutron"
# ironic_neutron_cleaning_network_uuid: "UUID for cleaning network in␣
↪neutron"
# ironic_neutron_inspection_network_uuid: "UUID for inspection network in␣
↪neutron"
# ironic_neutron_provisioning_network_name: "Name of provisioning network␣
↪in neutron"
# ironic_neutron_cleaning_network_name: "Name of cleaning network in␣
↪neutron"
# ironic_neutron_inspection_network_name: "Name of inspection network in␣
↪neutron"

# Integrated Openstack configuration
ironic_enabled_network_interfaces_list: "flat,noop{{ (ironic_neutron_
↪provisioning_network_uuid is defined) | ternary(',neutron','') }}"
ironic_default_network_interface: "{{ (ironic_neutron_provisioning_network_
↪uuid is defined) | ternary('neutron','flat') }}"
ironic_auth_strategy: keystone
ironic_dhcp_provider: "{{ (ironic_standalone | bool) | ternary('none',
↪'neutron') }}"
ironic_sync_power_state_interval: "{{ (ironic_standalone | bool) | ternary(
↪'-1', '60') }}"
ironic_db_connection_string: "mysql+pymysql://{{ ironic_galera_user }}:{{␣
↪ironic_container_mysql_password }}@{{ ironic_galera_address }}:{{ ironic_
↪galera_port }}/ironic{% if ironic_galera_use_ssl | bool %}&ssl_ca={{␣
↪ironic_galera_ssl_ca_cert }}{% endif %}"

# Ironic db tuning
ironic_db_max_overflow: 10
ironic_db_max_pool_size: 120
ironic_db_pool_timeout: 30

# Common configuration
ironic_node_name: ironic

# If you want to regenerate the ironic users SSH keys, on each run, set␣
↪this
# var to True. Otherwise keys will be generated on the first run and not
```

(continues on next page)

(continued from previous page)

```
# regenerated each run.
ironic_recreate_keys: False

ironic_tftp_server_address: "{{ ansible_host }}"

ironic_pip_packages:
  - cryptography
  - ironic
  - osprofiler
  - proliantutils
  - PyMySQL
  - pymemcache
  - pysnmp
  - python-dracclient
  - python-ilorest-library
  - python-ironicclient
  - python-memcached
  - python-scciclient
  - python-swiftclient
  - python-xclarityclient
  - sushy
  - systemd-python

ironic_inspector_pip_packages:
  - ironic-inspector
  - python-ironic-inspector-client

# Memcached override
ironic_memcached_servers: "{{ memcached_servers }}"

## Oslo Messaging Info
# RPC
ironic_oslomsg_rpc_host_group: "{{ oslomsg_rpc_host_group | default(
↪'rabbitmq_all') }}"
ironic_oslomsg_rpc_setup_host: "{{ (ironic_oslomsg_rpc_host_group in␣
↪groups) | ternary(groups[ironic_oslomsg_rpc_host_group][0], 'localhost')␣
↪}}"
ironic_oslomsg_rpc_transport: "{{ oslomsg_rpc_transport | default('rabbit
↪') }}"
ironic_oslomsg_rpc_servers: "{{ oslomsg_rpc_servers | default('127.0.0.1')␣
↪}}"
ironic_oslomsg_rpc_port: "{{ oslomsg_rpc_port | default('5672') }}"
ironic_oslomsg_rpc_use_ssl: "{{ oslomsg_rpc_use_ssl | default(False) }}"
ironic_oslomsg_rpc_userid: ironic
ironic_oslomsg_rpc_vhost: /ironic

# Notify
ironic_oslomsg_notify_host_group: "{{ oslomsg_notify_host_group | default(
↪'rabbitmq_all') }}"
ironic_oslomsg_notify_setup_host: "{{ (ironic_oslomsg_notify_host_group in␣
↪groups) | ternary(groups[ironic_oslomsg_notify_host_group][0], 'localhost
↪') }}"
ironic_oslomsg_notify_transport: "{{ oslomsg_notify_transport | default(
↪'rabbit') }}"
ironic_oslomsg_notify_servers: "{{ oslomsg_notify_servers | default('127.0.
↪0.1') }}"
```

(continues on next page)

(continued from previous page)

```
ironic_oslomsg_notify_port: "{{ oslomsg_notify_port | default('5672') }}"
ironic_oslomsg_notify_use_ssl: "{{ oslomsg_notify_use_ssl | default(False)␣
↪}}"
ironic_oslomsg_notify_userid: "{{ ironic_oslomsg_rpc_userid }}"
ironic_oslomsg_notify_password: "{{ ironic_oslomsg_rpc_password }}"
ironic_oslomsg_notify_vhost: "{{ ironic_oslomsg_rpc_vhost }}"

## (Qdrouterd) integration
# TODO(ansmith): Change structure when more backends will be supported
ironic_oslomsg_amqp1_enabled: "{{ ironic_oslomsg_rpc_transport == 'amqp' }}
↪"

ironic_optional_oslomsg_amqp1_pip_packages:
  - oslo.messaging[amqp1]

# Auth
ironic_service_user_name: "ironic"

# WSGI settings
ironic_wsgi_threads: 1
ironic_wsgi_processes_max: 16
ironic_wsgi_processes: "{{ [[(ansible_processor_vcpus//ansible_processor_
↪threads_per_core)|default(1), 1] | max * 2, ironic_wsgi_processes_max] |␣
↪min }}"
ironic_uwsgi_bind_address: "{{ openstack_service_bind_address | default('0.
↪0.0.0') }}"

### OpenStack Services to integrate with

# Glance
ironic_glance_auth_strategy: "{{ ironic_auth_strategy }}"
ironic_glance_service_project_name: "{{ glance_service_project_name |␣
↪default('service') }}"
ironic_glance_service_project_domain_id: "{{ glance_service_project_domain_
↪id | default('default') }}"
ironic_glance_keystone_auth_plugin: "{{ glance_keystone_auth_plugin |␣
↪default('password') }}"
ironic_glance_service_user_name: "{{ glance_service_user_name | default(
↪'glance') }}"
ironic_glance_service_user_domain_id: "{{ glance_service_user_domain_id |␣
↪default('default') }}"
ironic_glance_keystone_auth_url: "{{ keystone_service_internalurl |␣
↪default('http://localhost:5000/v3') }}"

# Neutron
ironic_neutron_auth_strategy: "{{ ironic_auth_strategy }}"

### Config Overrides
ironic_ironic_conf_overrides: {}
ironic_rootwrap_conf_overrides: {}
ironic_policy_overrides: {}
ironic_api_uwsgi_ini_overrides: {}

# pxe boot
ironic_pxe_append_params: "ipa-debug=1 systemd.journald.forward_to_
↪console=yes"
```

(continues on next page)

(continued from previous page)

```
ironic_api_init_config_overrides: {}
ironic_conductor_init_config_overrides: {}

# driver definitions
ironic_drivers_enabled:
  - agent_ipmitool
  - pxe_ipmitool

ironic_inspector_developer_mode: false
ironic_inspector_venv_python_executable: "{{ openstack_venv_python_
↪executable | default('python2') }}"

# System info
ironic_inspector_service_setup_host: "{{ openstack_service_setup_host |␣
↪default('localhost') }}"
ironic_inspector_lock_path: /var/lock/ironic-inspector

ironic_inspector_service_name: ironic-inspector
ironic_inspector_service_type: baremetal-introspection
ironic_inspector_service_description: "Ironic Baremetal Introspection␣
↪Service"
ironic_inspector_service_publicuri_proto: "{{ openstack_service_publicuri_
↪proto | default(ironic_service_proto) }}"
ironic_inspector_service_adminuri_proto: "{{ openstack_service_adminuri_
↪proto | default(ironic_service_proto) }}"
ironic_inspector_service_internaluri_proto: "{{ openstack_service_
↪internaluri_proto | default(ironic_service_proto) }}"
ironic_inspector_service_address: "{{ openstack_service_bind_address }}"
ironic_inspector_service_port: 5050
ironic_inspector_service_publicuri: "{{ ironic_inspector_service_publicuri_
↪proto }}://{{ external_lb_vip_address }}:{{ ironic_inspector_service_
↪port }}"
ironic_inspector_service_publicurl: "{{ ironic_inspector_service_publicuri␣
↪}}"
ironic_inspector_service_adminuri: "{{ ironic_inspector_service_adminuri_
↪proto }}://{{ internal_lb_vip_address }}:{{ ironic_inspector_service_
↪port }}"
ironic_inspector_service_adminurl: "{{ ironic_inspector_service_adminuri }}
↪"
ironic_inspector_service_internaluri: "{{ ironic_inspector_service_
↪internaluri_proto }}://{{ internal_lb_vip_address }}:{{ ironic_inspector_
↪service_port }}"
ironic_inspector_service_internalurl: "{{ ironic_inspector_service_
↪internaluri }}"
ironic_inspector_service_role_name: "admin"
ironic_inspector_service_project_name: "service"
ironic_inspector_service_in_ldap: False
ironic_inspector_service_domain_id: default

# Database
ironic_inspector_db_setup_host: "{{ openstack_db_setup_host | default(
↪'localhost') }}"
ironic_inspector_db_setup_python_interpreter: "{{ openstack_db_setup_
↪python_interpreter | default((ironic_inspector_db_setup_host ==
↪'localhost') | ternary(ansible_playbook_python, ansible_python[
↪'executable'])) }}"
```

(continues on next page)

(continued from previous page)

```
ironic_inspector_galera_address: "{{ galera_address | default('127.0.0.1')␣
↪}}"
ironic_inspector_galera_user: ironic-inspector
ironic_inspector_galera_database: ironic_inspector
ironic_inspector_galera_port: 3306

# Ironic db tuning
ironic_inspector_db_max_overflow: 10
ironic_inspector_db_max_pool_size: 120
ironic_inspector_db_pool_timeout: 30

ironic_inspector_pip_install_args: "{{ pip_install_options | default('') }}
↪"

# Ironic iPXE support
ironic_ipxe_enabled: False
ironic_ipxe_port: 8051
ironic_ipxe_proto: "http"

# Auth
ironic_inspector_service_user_name: "ironic_inspector"

### OpenStack Services to integrate with
# Ironic swift store information
ironic_inspector_swift_user_name: swift-inspector
ironic_inspector_swift_role_names:
  - _member_
  - swiftoperator

# Ironic inspector
ironic_inspector_enable_discovery: True
ironic_inspector_openstack_db_connection_string: "mysql+pymysql://{{␣
↪ironic_inspector_galera_user }}:{{ ironic_inspector_container_mysql_
↪password }}@{{ ironic_inspector_galera_address }}:{{ ironic_inspector_
↪galera_port }}/{{ ironic_inspector_galera_database }}"

# Ironic inspector dhcp
ironic_inspector_dhcp_pool_range: 192.168.0.51 192.168.0.150
ironic_inspector_dhcp_subnet: 192.168.0.0/22
ironic_inspector_dhcp_subnet_mask: 255.255.252.0
ironic_inspector_dhcp_gateway: 192.168.0.1
ironic_inspector_dhcp_nameservers: 192.168.0.1
ironic_inspector_dhcp_lease_time: 600

ironic_inspector_dhcp_type: dnsmasq # isc_dhcp
ironic_inspector_boot_mode: http #tftp
ironic_inspector_pxe_boot_mode: "{{ ironic_inspector_boot_mode }}"
ironic_inspector_httpboot_dir: /httpboot
ironic_inspector_tftpboot_dir: "{{ ironic_tftpd_root }}"

ironic_inspector_dhcp_interface: br-ironic
ironic_inspector_valid_interfaces: internal,public

### Config Overrides
ironic_inspector_conf_overrides: {}
ironic_inspector_rootwrap_conf_overrides: {}
```

(continues on next page)

(continued from previous page)

```
ironic_inspector_init_config_overrides: {}
# pxe boot
ironic_inspector_pxe_append_params: "ipa-debug=1 systemd.journald.forward_
↪to_console=yes" #ipa-inspection-collectors=default,logs,extra_hardware

ironic_inspector_pxe_filter: dnsmasq #iptables

ironic_inspector_oslomsg_rpc_host_group: "{{ oslomsg_rpc_host_group |␣
↪default('rabbitmq_all') }}"
ironic_inspector_oslomsg_rpc_setup_host: "{{ (ironic_oslomsg_rpc_host_
↪group in groups) | ternary(groups[ironic_oslomsg_rpc_host_group][0],
↪'localhost') }}"
ironic_inspector_oslomsg_rpc_transport: "{{ oslomsg_rpc_transport |␣
↪default('rabbit') }}"
ironic_inspector_oslomsg_rpc_servers: "{{ oslomsg_rpc_servers | default(
↪'127.0.0.1') }}"
ironic_inspector_oslomsg_rpc_port: "{{ oslomsg_rpc_port | default('5672') }
↪}"
ironic_inspector_oslomsg_rpc_use_ssl: "True"
ironic_inspector_oslomsg_rpc_userid: ironic
ironic_inspector_oslomsg_rpc_vhost: /ironic

ironic_inspector_oslomsg_notify_host_group: "{{ oslomsg_notify_host_group␣
↪| default('rabbitmq_all') }}"
ironic_inspector_oslomsg_notify_setup_host: "{{ (ironic_inspector_oslomsg_
↪notify_host_group in groups) | ternary(groups[ironic_inspector_oslomsg_
↪notify_host_group][0], 'localhost') }}"
ironic_inspector_oslomsg_notify_transport: "{{ oslomsg_notify_transport |␣
↪default('rabbit') }}"
ironic_inspector_oslomsg_notify_servers: "{{ oslomsg_notify_servers |␣
↪default('127.0.0.1') }}"
ironic_inspector_oslomsg_notify_port: "{{ oslomsg_notify_port | default(
↪'5672') }}"
ironic_inspector_oslomsg_notify_use_ssl: "False"
ironic_inspector_oslomsg_notify_userid: "{{ ironic_inspector_oslomsg_rpc_
↪userid }}"
ironic_inspector_oslomsg_notify_password: "{{ ironic_oslomsg_rpc_password }
↪}"
ironic_inspector_oslomsg_notify_vhost: "{{ ironic_inspector_oslomsg_rpc_
↪vhost }}"
ironic_inspector_optional_oslomsg_amqp1_pip_packages:
  - oslo.messaging[amqp1]
ironic_inspector_oslomsg_amqp1_enabled: True

ironic_inspector_ipa_initrd_name: ironic-deploy.initrd
ironic_inspector_ipa_kernel_name: ironic-deploy.kernel
```

CHAPTER

FOUR

DEPENDENCIES

This role needs pip >= 7.1 installed on the target host.

CHAPTER

FIVE

EXAMPLE PLAYBOOK

```
---
- name: Playbook for role testing
  hosts: localhost
  remote_user: root
  roles:
    - role: "os_ironic"
  vars:
    galera_root_user: root
  vars_prompt:
    - name: "galera_root_password"
      prompt: "What is galera_root_password?"
```

CHAPTER

SIX

TAGS

This role supports the `ironic-install` and `ironic-config` tags. Use the `ironic-install` tag to install and upgrade. Use the `ironic-config` tag to maintain configuration of the service.