OpenStack-Ansible Documentation: os_zun role

Release 0.1.0.dev210

OpenStack-Ansible Contributors

Oct 12, 2023

CONTENTS

1 Configuring the Compute (zun) service (optional) 1
- 1.1 Availability zones . . . 1
- 1.2 Block device tuning for Ceph (RBD) . . . 1
- 1.3 Config drive . . . 2
- 1.4 Libvirtd connectivity and authentication . . . 2
- 1.5 Multipath . . . 3
- 1.6 Shared storage and synchronized UID/GID . . . 3

2 Scenario - Using PowerVM Nova plugin 5
- 2.1 Prerequisites . . . 5
- 2.2 PowerVM usage . . . 5
- 2.3 Configuring storage . . . 5
- 2.4 Enabling VNC console . . . 6
- 2.5 Enabling configuration drive . . . 6
- 2.6 Enabling PowerVM RMC . . . 6

3 Default variables 7

4 Dependencies 17

5 Example playbook 19

6 Tags 21

7 CPU platform compatibility 23

8 Compute driver compatibility 25

CHAPTER

ONE

CONFIGURING THE COMPUTE (ZUN) SERVICE (OPTIONAL)

The Compute service (zun) handles the creation of virtual machines within an OpenStack environment. Many of the default options used by OpenStack-Ansible are found within `defaults/main.yml` within the zun role.

1.1 Availability zones

Deployers with multiple availability zones can set the `zun_default_schedule_zone` Ansible variable to specify an availability zone for new requests. This is useful in environments with different types of hypervisors, where builds are sent to certain hardware types based on their resource requirements.

For example, if you have servers running on two racks without sharing the PDU. These two racks can be grouped into two availability zones. When one rack loses power, the other one still works. By spreading your containers onto the two racks (availability zones), you will improve your service availability.

1.2 Block device tuning for Ceph (RBD)

Enabling Ceph and defining `zun_libvirt_images_rbd_pool` changes two libvirt configurations by default:

- hw_disk_discard: `unmap`
- disk_cachemodes: `network=writeback`

Setting `hw_disk_discard` to `unmap` in libvirt enables discard (sometimes called TRIM) support for the underlying block device. This allows reclaiming of unused blocks on the underlying disks.

Setting `disk_cachemodes` to `network=writeback` allows data to be written into a cache on each change, but those changes are flushed to disk at a regular interval. This can increase write performance on Ceph block devices.

You have the option to customize these settings using two Ansible variables (defaults shown here):

```
zun_libvirt_hw_disk_discard: 'unmap'
zun_libvirt_disk_cachemodes: 'network=writeback'
```

You can disable discard by setting `zun_libvirt_hw_disk_discard` to `ignore`. The `zun_libvirt_disk_cachemodes` can be set to an empty string to disable `network=writeback`.

The following minimal example configuration sets zun to use the `ephemeral-vms` Ceph pool. The following example uses cephx authentication, and requires an existing `cinder` account for the `ephemeral-vms` pool:

```
zun_libvirt_images_rbd_pool: ephemeral-vms
ceph_mons:
  - 172.29.244.151
  - 172.29.244.152
  - 172.29.244.153
```

If you have a different Ceph username for the pool, use it as:

```
cinder_ceph_client: <ceph-username>
```

- The Ceph documentation for OpenStack has additional information about these settings.
- OpenStack-Ansible and Ceph Working Example

1.3 Config drive

By default, OpenStack-Ansible does not configure zun to force config drives to be provisioned with every instance that zun builds. The metadata service provides configuration information that is used by `cloud-init` inside the instance. Config drives are only necessary when an instance does not have `cloud-init` installed or does not have support for handling metadata.

A deployer can set an Ansible variable to force config drives to be deployed with every virtual machine:

```
zun_force_config_drive: True
```

Certain formats of config drives can prevent instances from migrating properly between hypervisors. If you need forced config drives and the ability to migrate instances, set the config drive format to `vfat` using the `zun_zun_conf_overrides` variable:

```
zun_zun_conf_overrides:
  DEFAULT:
    config_drive_format: vfat
    force_config_drive: True
```

1.4 Libvirtd connectivity and authentication

By default, OpenStack-Ansible configures the libvirt daemon in the following way:

- TLS connections are enabled
- TCP plaintext connections are disabled
- Authentication over TCP connections uses SASL

You can customize these settings using the following Ansible variables:

```
# Enable libvirtd's TLS listener
zun_libvirtd_listen_tls: 1

# Disable libvirtd's plaintext TCP listener
zun_libvirtd_listen_tcp: 0
```

(continues on next page)

(continued from previous page)

```
# Use SASL for authentication
zun_libvirtd_auth_tcp: sasl
```

1.5 Multipath

Nova supports multipath for iSCSI-based storage. Enable multipath support in zun through a configuration override:

```
zun_zun_conf_overrides:
  libvirt:
      iscsi_use_multipath: true
```

1.6 Shared storage and synchronized UID/GID

Specify a custom UID for the zun user and GID for the zun group to ensure they are identical on each host. This is helpful when using shared storage on Compute nodes because it allows instances to migrate without filesystem ownership failures.

By default, Ansible creates the zun user and group without specifying the UID or GID. To specify custom values for the UID or GID, set the following Ansible variables:

```
zun_system_user_uid = <specify a UID>
zun_system_group_gid = <specify a GID>
```

Warning: Setting this value after deploying an environment with OpenStack-Ansible can cause failures, errors, and general instability. These values should only be set once before deploying an OpenStack environment and then never changed.

CHAPTER

TWO

SCENARIO - USING POWERVM NOVA PLUGIN

2.1 Prerequisites

In order to use the PowerVM OpenStack drivers with OpenStack-Ansible (OSA), the following pre-requisites must be fulfilled:

- At least one of the repo-build servers must be ppc64le. Can mix and match repo-build servers between x86 and ppc64le.
- The compute nodes should be pre-configured for PowerVM with the NovaLink feature.
- The NovaLink Management VM needs at least one direct attach I/O card. OpenStack Ansible is currently able to deploy the PowerVM drivers when paired with the Open vSwitch agent. The traditional PowerVM Shared Ethernet Adapter networking agent is not yet supported.
- The network topology on the NovaLink must match a supported OpenStack Ansible network configuration.

2.2 PowerVM usage

The Compute driver for OpenStack-Ansible should automatically detect that it is of type PowerVM. If the user has specified a specific compute type, that is applicable to the whole cloud. It is advised that the you allow OSA to detect the appropriate compute node type.

The full set of configuration options for the PowerVM driver can be found in the `zun-powervm` usage.

2.3 Configuring storage

There are various storage back ends available for PowerVM such as local disk and shared storage pools. For example, to enable local disk storage backed by a logical volume group, you can set:

```
zun_zun_conf_overrides:
  powervm:
    disk_driver: localdisk
    volume_group_name: <<VOLUME GROUP NAME>>
```

To enable iSCSI as the volume attachment type, you can set the `volume_adapter` setting:

```
zun_zun_conf_overrides:
  powervm:
    volume_adapter: iscsi
```

The default volume attachment type for PowerVM is fibre channel.

2.4 Enabling VNC console

PowerVM only supports connecting to instance consoles over VNC. As OpenStack-Ansible defaults to Spice console, you must set the `zun_console_type` variable to enable NoVNC:

```
zun_console_type: novnc
```

2.5 Enabling configuration drive

By default, PowerVM uses configuration drives to provide configuration information to instances built by zun. To enable this support in OpenStack-Ansible, you can set the `zun_force_config_drive` variable as documented in the zun configuration instructions.

Additionally, you can enable flat network injection by using the `zun_zun_conf_overrides` variable:

```
zun_zun_conf_overrides:
  DEFAULT:
    flat_injected: True
```

2.6 Enabling PowerVM RMC

To enable PowerVM RMC, IPv4/IPv6 dual-stack mode must be enabled. To do this, you must set `use_ipv6` using the `zun_zun_conf_overrides` variable:

```
zun_zun_conf_overrides:
  DEFAULT:
    use_ipv6: True
```

tags
: openstack, zun, cloud, ansible

category
: *nix

This role will install the following Systemd services:

- zun-server
- zun-compute

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

CHAPTER

THREE

DEFAULT VARIABLES

```
# Enable/Disable barbican configurations
zun_barbican_enabled: "{{ (groups['barbican_all'] is defined) and (groups[
↪'barbican_all'] | length > 0) }}"
# Enable/Disable designate configurations
zun_designate_enabled: "{{ (groups['designate_all'] is defined) and (groups[
↪'designate_all'] | length > 0) }}"
# Notification topics for designate.
zun_notifications_designate: notifications_designate
# Enable/Disable ceilometer configurations
zun_ceilometer_enabled: "{{ (groups['ceilometer_all'] is defined) and (groups[
↪'ceilometer_all'] | length > 0) }}"

## Verbosity Options
debug: False

#python venv executable
zun_venv_python_executable: "{{ openstack_venv_python_executable | default(
↪'python3') }}"

# Set the host which will execute the shade modules
# for the service setup. The host must already have
# clouds.yaml properly configured.
zun_service_setup_host: "{{ openstack_service_setup_host | default('localhost
↪') }}"
zun_service_setup_host_python_interpreter: "{{ openstack_service_setup_host_
↪python_interpreter | default((zun_service_setup_host == 'localhost') |␣
↪ternary(ansible_playbook_python, ansible_facts['python']['executable'])) }}"

# Set the package install state for distribution packages
# Options are 'present' and 'latest'
zun_package_state: "{{ package_state | default('latest') }}"

zun_git_repo: https://opendev.org/openstack/zun
zun_git_install_branch: master

zun_kuryr_git_repo: https://opendev.org/openstack/kuryr-libnetwork
zun_kuryr_git_install_branch: master
```

(continues on next page)

(continued from previous page)

```
# This is only required until kuryr-libnetwork depends upon a version of␣
↪kuryr-lib
# which includes https://review.opendev.org/c/openstack/kuryr/+/764908
zun_kuryr_lib_git_repo: https://opendev.org/openstack/kuryr
zun_kuryr_lib_git_install_branch: master

zun_upper_constraints_url: "{{ requirements_git_url | default('https://
↪releases.openstack.org/constraints/upper/' ~ requirements_git_install_
↪branch | default('master')) }}"
zun_git_constraints:
  - "--constraint {{ zun_upper_constraints_url }}"

zun_pip_install_args: "{{ pip_install_options | default('') }}"

# Name of the virtual env to deploy into
zun_venv_tag: "{{ venv_tag | default('untagged') }}"
zun_bin: "/openstack/venvs/zun-{{ zun_venv_tag }}/bin"

zun_fatal_deprecations: False

## Zun user information
zun_system_user_name: zun
zun_system_group_name: zun
zun_system_shell: /bin/false
zun_system_comment: zun system user
zun_system_home_folder: "/var/lib/{{ zun_system_user_name }}"
zun_system_slice_name: zun
zun_log_dir: "/var/log/zun"

zun_lock_dir: "{{ openstack_lock_dir | default('/run/lock') }}"

## Kuryr user information
zun_kuryr_system_user_name: kuryr
zun_kuryr_system_group_name: kuryr
zun_kuryr_system_shell: /bin/false
zun_kuryr_system_comment: kuryr system user
zun_kuryr_system_home_folder: "/var/lib/{{ zun_kuryr_system_user_name }}"
zun_kuryr_log_dir: "/var/log/kuryr"

## Docker setup information
zun_docker_package_version: "{{ _zun_docker_package_version }}"
zun_containerd_package_version: "{{ _zun_containerd_package_version }}"
zun_kata_package_version: "3.1.0"
zun_kata_package_source: "https://github.com/kata-containers/kata-containers/
↪releases/download/{{ zun_kata_package_version }}/kata-static-{{ zun_kata_
↪package_version }}-x86_64.tar.xz"
zun_kata_package_checksum:␣
↪sha256:452cc850e021539c14359d016aba18ddba128f59aa9ab637738296d9b5cd78a0
zun_kata_enabled: "True"
```

(continues on next page)

(continued from previous page)

```
# Set a list of users that are permitted to execute the docker binary.
zun_docker_users:
  - "{{ zun_system_user_name }}"
  - "{{ zun_kuryr_system_user_name }}"

# Set the docker api version. The default is false, which will result in no
# option being set in config for api servers. On compute hosts the docker api
# version will be used as determined by the client version information.
zun_docker_api_version: false

# Set the address for Docker to bind to. Used by the wsproxy console forwarder
zun_docker_bind_host: "{{ openstack_service_bind_address | default('0.0.0.0')␣
↪}}"
zun_docker_bind_port: 2375

# Should Docker image cache data be periodically cleaned up?
zun_docker_prune_images: False

# Time period for which to clean up old Docker data. The options are hour,␣
↪day,
# month, or year. (string value)
zun_docker_prune_frequency: hour

## Manually specified zun UID/GID
# Deployers can specify a UID for the zun user as well as the GID for the
# zun group if needed. This is commonly used in environments where shared
# storage is used, such as NFS or GlusterFS, and zun UID/GID values must be
# in sync between multiple servers.
#
# WARNING: Changing these values on an existing deployment can lead to
#          failures, errors, and instability.
#
# zun_system_user_uid = <UID>
# zun_system_group_gid = <GID>

## Database info
zun_db_setup_host: "{{ openstack_db_setup_host | default('localhost') }}"
zun_db_setup_python_interpreter: "{{ openstack_db_setup_python_interpreter |␣
↪default((zun_db_setup_host == 'localhost') | ternary(ansible_playbook_
↪python, ansible_facts['python']['executable'])) }}"
zun_galera_address: "{{ galera_address | default('127.0.0.1') }}"
zun_galera_user: zun
zun_galera_database: zun
zun_db_max_overflow: "{{ openstack_db_max_overflow | default('50') }}"
zun_db_max_pool_size: "{{ openstack_db_max_pool_size | default('5') }}"
zun_db_pool_timeout: "{{ openstack_db_pool_timeout | default('30') }}"
zun_db_connection_recycle_time: "{{ openstack_db_connection_recycle_time |␣
↪default('600') }}"
```

(continues on next page)

(continued from previous page)

```
# Toggle whether zun connects via an encrypted connection
zun_galera_use_ssl: "{{ galera_use_ssl | default(False) }}"
# The path where to store the database server CA certificate
zun_galera_ssl_ca_cert: "{{ galera_ssl_ca_cert | default('') }}"
zun_galera_port: "{{ galera_port | default('3306') }}"

## RabbitMQ info

## Configuration for RPC communications
zun_rpc_thread_pool_size: 64
zun_rpc_conn_pool_size: 30
zun_rpc_response_timeout: 60

## Oslo Messaging info

# RPC
zun_oslomsg_rpc_host_group: "{{ oslomsg_rpc_host_group | default('rabbitmq_all
↪') }}"
zun_oslomsg_rpc_setup_host: "{{ (zun_oslomsg_rpc_host_group in groups) |␣
↪ternary(groups[zun_oslomsg_rpc_host_group][0], 'localhost') }}"
zun_oslomsg_rpc_transport: "{{ oslomsg_rpc_transport | default('rabbit') }}"
zun_oslomsg_rpc_servers: "{{ oslomsg_rpc_servers | default('127.0.0.1') }}"
zun_oslomsg_rpc_port: "{{ oslomsg_rpc_port | default('5672') }}"
zun_oslomsg_rpc_use_ssl: "{{ oslomsg_rpc_use_ssl | default(False) }}"
zun_oslomsg_rpc_userid: zun
zun_oslomsg_rpc_vhost: /zun
zun_oslomsg_rpc_ssl_version: "{{ oslomsg_rpc_ssl_version | default('TLSv1_2')␣
↪}}"
zun_oslomsg_rpc_ssl_ca_file: "{{ oslomsg_rpc_ssl_ca_file | default('') }}"

# Notify
zun_oslomsg_notify_host_group: "{{ oslomsg_notify_host_group | default(
↪'rabbitmq_all') }}"
zun_oslomsg_notify_setup_host: "{{ (zun_oslomsg_notify_host_group in groups)␣
↪| ternary(groups[zun_oslomsg_notify_host_group][0], 'localhost') }}"
zun_oslomsg_notify_transport: "{{ oslomsg_notify_transport | default('rabbit
↪') }}"
zun_oslomsg_notify_servers: "{{ oslomsg_notify_servers | default('127.0.0.1')␣
↪}}"
zun_oslomsg_notify_port: "{{ oslomsg_notify_port | default('5672') }}"
zun_oslomsg_notify_use_ssl: "{{ oslomsg_notify_use_ssl | default(False) }}"
zun_oslomsg_notify_userid: "{{ zun_oslomsg_rpc_userid }}"
zun_oslomsg_notify_password: "{{ zun_oslomsg_rpc_password }}"
zun_oslomsg_notify_vhost: "{{ zun_oslomsg_rpc_vhost }}"
zun_oslomsg_notify_ssl_version: "{{ oslomsg_notify_ssl_version | default(
↪'TLSv1_2') }}"
zun_oslomsg_notify_ssl_ca_file: "{{ oslomsg_notify_ssl_ca_file | default('') }
↪}"
```

(continues on next page)

(continued from previous page)

```
# If this is not set, then the playbook will try to guess it.
#zun_virt_type: kvm

## Zun Auth
zun_service_region: "{{ service_region | default('RegionOne') }}"
zun_service_project_name: "service"
zun_service_project_domain_id: default
zun_service_user_domain_id: default
zun_service_user_name: "zun"
zun_service_role_names:
  - admin
  - service
zun_service_token_roles:
  - service
zun_service_token_roles_required: "{{ openstack_service_token_roles_required␣
↪| default(True) }}"

## Zun Auth for kuryr
zun_kuryr_service_username: kuryr

## Keystone authentication middleware
zun_keystone_auth_plugin: password

## Zun WebSocket Proxy
zun_wsproxy_proto: "{{ (openstack_service_publicuri_proto | default('http')␣
↪== 'https') | ternary('wss', 'ws') }}"
zun_wsproxy_port: 6784
zun_wsproxy_host: "{{ openstack_service_bind_address | default('0.0.0.0') }}"
zun_wsproxy_base_uri: "{{ zun_wsproxy_proto }}://{{ external_lb_vip_address }}
↪:{{ zun_wsproxy_port }}"

## Zun v1
zun_service_name: zun
zun_service_type: container
zun_service_proto: http
zun_service_publicuri_proto: "{{ openstack_service_publicuri_proto |␣
↪default(zun_service_proto) }}"
zun_service_adminuri_proto: "{{ openstack_service_adminuri_proto |␣
↪default(zun_service_proto) }}"
zun_service_internaluri_proto: "{{ openstack_service_internaluri_proto |␣
↪default(zun_service_proto) }}"
zun_service_address: "{{ openstack_service_bind_address | default('0.0.0.0') }
↪}"
zun_service_port: 9517
zun_kuryr_service_address: 127.0.0.1
zun_kuryr_service_port: 23750
zun_service_description: "Zun Compute Service"
zun_service_publicuri: "{{ zun_service_publicuri_proto }}://{{ external_lb_
↪vip_address }}:{{ zun_service_port }}"
```

(continues on next page)

(continued from previous page)

```
zun_service_publicurl: "{{ zun_service_publicuri }}"
zun_service_adminuri: "{{ zun_service_adminuri_proto }}://{{ internal_lb_vip_
↪address }}:{{ zun_service_port }}"
zun_service_adminurl: "{{ zun_service_adminuri }}"
zun_service_internaluri: "{{ zun_service_internaluri_proto }}://{{ internal_
↪lb_vip_address }}:{{ zun_service_port }}"
zun_service_internalurl: "{{ zun_service_internaluri }}"
zun_service_endpoint_type: internalURL

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

## General Zun configuration
# Select between the 'runc' or 'kata' runtime
zun_container_runtime: runc

# If ``zun_osapi_compute_workers`` is unset the system will use half the number␣
↪of available VCPUS to
# compute the number of api workers to use.
# zun_osapi_compute_workers: 16

# If ``zun_conductor_workers`` is unset the system will use half the number of␣
↪available VCPUS to
# compute the number of api workers to use.
# zun_conductor_workers: 16

# If ``zun_metadata_workers`` is unset the system will use half the number of␣
↪available VCPUS to
# compute the number of api workers to use.
# zun_metadata_workers: 16

## Cap the maximun number of threads / workers when a user value is␣
↪unspecified.
zun_api_threads_max: 16
zun_api_threads: "{{ [[(ansible_facts['processor_vcpus']//ansible_facts[
↪'processor_threads_per_core'])|default(1), 1] | max * 2, zun_api_threads_
↪max] | min }}"

zun_service_in_ldap: "{{ service_ldap_backend_enabled | default(False) }}"

zun_scheduler_default_filters: >-
  AvailabilityZoneFilter,
  ComputeFilter
zun_scheduler_available_filters: zun.scheduler.filters.all_filters
zun_scheduler_driver: filter_scheduler
```

(continues on next page)

(continued from previous page)

```
## uWSGI setup
zun_wsgi_threads: 1
zun_wsgi_processes_max: 16
zun_wsgi_processes: "{{ [[ansible_facts['processor_vcpus']|default(1), 1] |␣
↪max * 2, zun_wsgi_processes_max] | min }}"

## Service Name-Group Mapping
zun_services:
  kuryr-libnetwork:
    group: zun_compute
    service_name: kuryr-libnetwork
    condition: "{{ inventory_hostname in groups['zun_compute'] }}"
    init_config_overrides: "{{ zun_kuryr_init_defaults | combine(zun_kuryr_
↪init_overrides, recursive=True) }}"
    start_order: 3
    wsgi_app: True
    wsgi: kuryr_libnetwork.server:app
    uwsgi_bind_address: "{{ zun_kuryr_service_address }}"
    uwsgi_port: "{{ zun_kuryr_service_port }}"
    uwsgi_overrides: "{{ zun_kuryr_uwsgi_conf_overrides }}"
    uwsgi_uid: "{{ zun_kuryr_system_user_name }}"
    uwsgi_guid: "{{ zun_kuryr_system_group_name }}"
  zun-api:
    group: zun_api
    service_name: zun-api
    init_config_overrides: "{{ zun_api_init_overrides }}"
    start_order: 1
    wsgi_app: True
    wsgi_path: "{{ zun_bin }}/zun-api-wsgi"
    uwsgi_bind_address: "{{ zun_service_address }}"
    uwsgi_port: "{{ zun_service_port }}"
    uwsgi_overrides: "{{ zun_uwsgi_conf_overrides }}"
    uwsgi_uid: "{{ zun_system_user_name }}"
    uwsgi_guid: "{{ zun_system_group_name }}"
  zun-compute:
    group: zun_compute
    service_name: zun-compute
    init_config_overrides: "{{ zun_compute_init_overrides }}"
    start_order: 5
    execstarts: "{{ zun_bin }}/zun-compute --config-dir /etc/zun"
  zun-wsproxy:
    group: zun_api
    service_name: zun-wsproxy
    init_config_overrides: "{{ zun_wsproxy_init_overrides }}"
    start_order: 2
    execstarts: "{{ zun_bin }}/zun-wsproxy --config-dir /etc/zun"
  zun-docker-cleanup:
    group: zun_compute
    service_name: zun-docker-cleanup
```

(continues on next page)

(continued from previous page)

```
    init_config_overrides: "{{ zun_docker_cleanup_init_overrides }}"
    start_order: 6
    execstarts: "{{ zun_bin }}/zun-docker-cleanup"
    timer:
      state: started
      options:
        OnBootSec: 30min
        OnCalendar: "{{ (zun_docker_prune_frequency == 'day') | ternary('daily
↪', zun_docker_prune_frequency+'ly') }}"
        Persistent: true
  docker:
    group: zun_compute
    service_name: docker
    init_config_overrides: {}
    start_order: 4
    systemd_overrides_only: True
    systemd_overrides: "{{ zun_docker_init_defaults | combine(zun_docker_init_
↪overrides, recursive=True) }}"

# Common pip packages
zun_pip_packages:
  - "git+{{ zun_git_repo }}@{{ zun_git_install_branch }}#egg=zun"
  - "git+{{ zun_kuryr_lib_git_repo }}@{{ zun_kuryr_lib_git_install_branch }}
↪#egg=kuryr-lib"
  - "git+{{ zun_kuryr_git_repo }}@{{ zun_kuryr_git_install_branch }}
↪#egg=kuryr-libnetwork"
  - oslo_rootwrap
  - osprofiler
  - python-memcached
  - pymemcache
  - python-zunclient
  - pymysql
  - systemd-python

## (Qdrouterd) integration
# TODO(ansmith): Change structure when more backends will be supported
zun_oslomsg_amqp1_enabled: "{{ zun_oslomsg_rpc_transport == 'amqp' }}"

zun_memcached_servers: "{{ memcached_servers }}"

zun_optional_oslomsg_amqp1_pip_packages:
  - oslo.messaging[amqp1]

## Default service options used within all systemd unit files.
zun_service_defaults: {}

## Tunable overrides for services
zun_zun_conf_overrides: {}
zun_rootwrap_conf_overrides: {}
```

(continues on next page)

(continued from previous page)

```
zun_kuryr_conf_overrides: {}
zun_docker_config_overrides: {}
zun_kuryr_config_overrides: {}
zun_uwsgi_conf_overrides: {}
zun_kuryr_uwsgi_conf_overrides:
  uwsgi:
    pyargv: --config-file /etc/kuryr/kuryr.conf

## Default zun+kuryr options used within the systemd unit file.
zun_kuryr_init_defaults:
  Unit:
    Before: docker.service
    After: network-online.target
    Wants: network-online.target
  Service:
    CapabilityBoundingSet: CAP_NET_ADMIN
    AmbientCapabilities: CAP_NET_ADMIN
    Group: "{{ zun_kuryr_system_group_name }}"
    User: "{{ zun_kuryr_system_user_name }}"

## Default zun+docker options used within the systemd unit file.
zun_docker_init_defaults:
  Service:
    ExecStart:
      - ""
      - "/usr/bin/dockerd --group {{ zun_system_group_name }} -H tcp://{{ zun_
↪docker_bind_host }}:{{ zun_docker_bind_port }} -H unix:///var/run/docker.
↪sock --cluster-store etcd://{% for item in groups['zun_api'] %}{{␣
↪hostvars[item]['management_address'] }}:2379{% if not loop.last %},{% endif
↪%}{% endfor %}{% if zun_kata_enabled %} --add-runtime kata=/opt/kata/bin/
↪kata-runtime{% endif %}"

## Tunable overrides for service unit files.
zun_api_paste_ini_overrides: {}
zun_api_init_overrides: {}
zun_wsproxy_init_overrides: {}
zun_compute_init_overrides: {}
zun_kuryr_init_overrides: {}
zun_docker_init_overrides: {}
zun_docker_cleanup_init_overrides: {}
zun_policy_overrides: {}
```

CHAPTER

FOUR

DEPENDENCIES

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

CHAPTER

FIVE

EXAMPLE PLAYBOOK

```
---

- name: Gather zun facts
  hosts: zun_all
  gather_facts: True
  tags:
    - always

- name: Install zun services
  hosts: zun_all
  gather_facts: False
  serial:
    - 1
    - "100%"
  user: root
  environment: "{{ deployment_environment_variables | default({}) }}"
  tags:
    - zun
  roles:
   - role: "os_zun"
```

CHAPTER

SIX

TAGS

This role supports two tags: `zun-install` and `zun-config`

The `zun-install` tag can be used to install and upgrade.

The `zun-config` tag can be used to manage configuration.

CHAPTER

SEVEN

CPU PLATFORM COMPATIBILITY

This role supports multiple CPU architecture types. At least one repo_build node must exist for each CPU type that is in use in the deployment.

Currently supported CPU architectures:

- x86_64 / amd64
- ppc64le

At this time, ppc64le is only supported for the Compute node type. It can not be used to manage the OpenStack-Ansible management nodes.

CHAPTER

EIGHT

COMPUTE DRIVER COMPATIBILITY

This role supports multiple zun compute driver types. The following compute drivers are supported:

- libvirt (default)
- ironic
- lxd (via zun-lxd)
- powervm (via zun-powervm)

The driver type is automatically detected by the OpenStack Ansible Nova role for the following compute driver types:

- libvirt (kvm / qemu)
- powervm

Any mix and match of compute node types can be used for those platforms, except for ironic.

If using the lxd driver, the compute type must be specified using the `zun_virt_type` variable.

The `zun_virt_type` may be set in `/etc/openstack_deploy/user_variables.yml`, for example:

```
zun_virt_type: lxd
```

You can set `zun_virt_type` per host by using `host_vars` in `/etc/openstack_deploy/openstack_user_config.yml`. For example:

```
compute_hosts:
 aio1:
   ip: 172.29.236.100
   host_vars:
     zun_virt_type: lxd
```

If `zun_virt_type` is set in `/etc/openstack_deploy/user_variables.yml`, all nodes in the deployment are set to that hypervisor type. Setting `zun_virt_type` in both `/etc/openstack_deploy/user_variables.yml` and `/etc/openstack_deploy/openstack_user_config.yml` will always result in the value specified in `/etc/openstack_deploy/user_variables.yml` being set on all hosts.