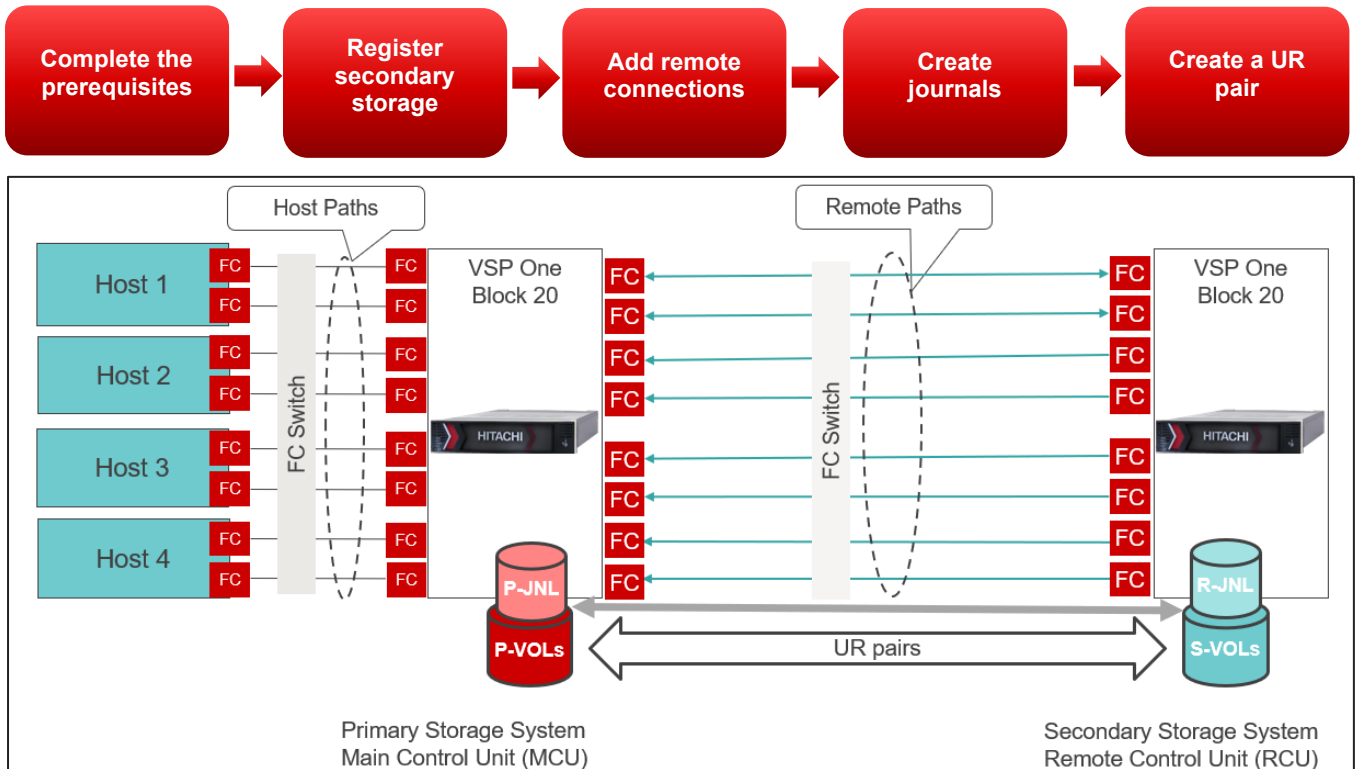


Deploy Universal Replicator using Ansible

Universal Replicator (UR) software is an asynchronous data replication solution for business continuity and disaster recovery across distances. It copies data asynchronously from primary volumes (P-VOL) at the main data center to secondary volumes (S-VOL) at a remote site.

Hitachi Virtual Storage Platform One Block (VSP One Block) Storage Modules are a collection of Ansible modules designed to automate provisioning, configuration, and replication tasks on VSP One Block storage systems. These modules integrate seamlessly with Red Hat Ansible, helping to simplify day-to-day storage operations through consistent and repeatable automation.

Get started with these simple steps on how you can deploy UR using Ansible:



Step 1: Complete the prerequisites

- Provision the data volumes and volumes for journals on the designated pools in both storage systems.
- Establish Fibre Channel (FC) zoning or IP network connectivity between the Main Control Unit (MCU) and Remote Control Unit (RCU).
- Make sure you have sufficient privileges to run replication commands in Ansible.
- Install the Ansible collection for VSP One Block:

[Ansible Galaxy](#). Search for namespace Hitachi Vantara or modules `vspone_block`. Install the adapter following the instructions posted to Ansible Galaxy.

This downloads and installs the collection from galaxy.ansible.com under:

```
~/.ansible/collections/ansible_collections/hitachivantara/vspone_block/
```

Navigate to the `playbooks/` directory inside the collection. This directory includes:

- `ansible_vault_vars/`: Contains the `ansible_vault_storage_var.yml` file. Update this file with the MCU and RCU credentials:

```
storage_serial: <primarySerialNumber>
storage_address: <StorageManagementAddress>
vault_storage_username: <username>
vault_storage_secret: <password>
```

```
secondary_storage_serial: <secondarySerialNumber>
secondary_storage_address: <StorageManagementAddress>
vault_secondary_storage_username: <username>
vault_secondary_storage_secret: <password>
```

- `vsp_direct/`: Contains sample playbooks. Copy the relevant task blocks from the master `entity.yml` or `entity_facts.yml` files to create your playbooks.

Use Ansible Vault to encrypt this file and protect sensitive credentials such as usernames and passwords. Run the `ansible-vault encrypt ansible_vault_storage_var.yml` command and enter the vault password when prompted. While running the playbook, use the `--ask-vault-pass` additional command line option to supply the vault password.

Step 2: Register secondary storage

To enable UR and perform remote replication, the primary and secondary VSP storage systems must be registered with each other. This step establishes connectivity between the MCU and RCU.

1. Copy the sample playbook `remote_storage_registration.yml`.
2. In the playbook, load the storage credentials from the `ansible_vault_storage_var.yml` variable file using the `vars_files` option.
3. Use the `hitachivantara.vspone_block.vsp.hv_remote_storage_registration` module to register the secondary storage system.
4. Set the parameter `is_mutual_discovery: true` to allow both systems to auto-discover each other.
5. Make sure that both VSP storage systems are reachable from the Ansible control host.

The following is a sample playbook to register the secondary/remote storage system:

```
- name: Remote Storage Registration/ Unregistration Playbook
  hosts: localhost
  gather_facts: false

  vars_files:
  - ../ansible_vault_vars/ansible_vault_storage_var.yml

  vars:
  connection_info:
    address: "{{ storage_address }}"
    username: "{{ vault_storage_username }}"
    password: "{{ vault_storage_secret }}"
  secondary_connection_info:
    address: "{{ secondary_storage_address }}"
    username: "{{ vault_secondary_storage_username }}"
    password: "{{ vault_secondary_storage_secret }}"

  tasks:
  #####
  # Task Name: Register Remote Storage
  #####

  - name: Register Remote Storage
    hitachivantara.vspone_block.vsp.hv_remote_storage_registration:
      connection_info: "{{ connection_info }}"
      secondary_connection_info: "{{ secondary_connection_info }}"
      state: present
      spec:
        rest_server_ip: "<StorageManagementAddress>"
        is_mutual_discovery: true
    register: result

  - name: Debug result
    debug:
      var: result
```

Step 3: Add remote connections

After registering the secondary storage system, define the remote I/O paths between the primary and secondary VSP storage systems. This enables the transfer of replication data between specific controller ports on both systems.

1. Identify the local and remote port labels for replication. For example: CL7-A, CL8-A.
2. Assign a `path_group_id` to logically group the paths. For most configurations, 0 is used as the default group.
3. Obtain the serial number of the secondary storage system from the system GUI or through the API.
4. Define the required number of remote paths and specify connection parameters such as I/O timeout and round-trip latency.
5. Run the playbook using the `hv_remote_connection` module to define the remote I/O paths between the primary and secondary storage systems. The paths are grouped logically under the specified `path_group_id`.
6. Repeat step 5 and run a similar playbook from the secondary storage system to establish a bi-directional remote connectivity.

The following is a sample playbook referencing `remote_connection.yml` to add a remote connection with two remote paths under a single path group:

```
- name: Remote Connection Creation Playbook
  hosts: localhost
  gather_facts: false

  vars_files:
    - ../ansible_vault_vars/ansible_vault_storage_var.yml

  vars:
    connection_info:
      address: "{{ storage_address }}"
      username: "{{ vault_storage_username }}"
      password: "{{ vault_storage_secret }}"

  tasks:
    #####
    # Task 1: Create a new remote connection
    #####
    - name: Create a new remote connection
      hitachivantara.vspone_block.vsp.hv_remote_connection:
        connection_info: "{{ connection_info }}"
        state: present
        spec:
          path_group_id: 0
          remote_storage_serial_number: "810002"
          remote_paths:
            - local_port: "CL7-A"
              remote_port: "CL7-A"
            - local_port: "CL8-A"
              remote_port: "CL8-A"
          min_remote_paths: 1
          remote_io_timeout_in_sec: 15
          round_trip_in_msec: 1
        register: result

    - name: Debug result
      debug:
        var: result
```

Step 4: Create journals

For UR operations, journal volumes must be configured on both the primary and secondary storage systems. Updates to the P-VOL are copied to the master journal volume in the primary storage system. Master journal data is copied to the restore journal volume on the secondary storage system. A journal group consists of one or more data volumes and the associated journal volume.

1. Identify or create a target pool on each storage system to hold journal volumes.
2. Select an unused LDEV ID, for example, 600, for each journal volume you want to create.
3. Use the `hitachivantara.vspone_block.vsp.hv_ldev` module to create a volume in the selected pool.
4. Use the `hitachivantara.vspone_block.vsp.hv_journal_volume` module to assign the volume to a journal ID.
5. Repeat the procedure on the secondary system using a matching journal ID and an associated LDEV ID.

The following is the sample playbook referencing `ldev.yml` and `journal_volume.yml` to create the volume first and then define journal on that volume:

```
- name: Ansible Playbook for Journal Volume Creation
  hosts: localhost
  gather_facts: false

  vars_files:
    - ../ansible_vault_vars/ansible_vault_storage_var.yml

  vars:
    connection_info:
      address: "{{ storage_address }}"
      username: "{{ vault_storage_username }}"
      password: "{{ vault_storage_secret }}"

  tasks:
    #####
    # Task Name: Create journal LDEV
    #####
    - name: Create journal ldevs
      hitachivantara.vspone_block.vsp.hv_ldev:
        connection_info: "{{ connection_info }}"
        state: "present"
        spec:
          ldev_id: 600
          pool_id: 1
          size: "100GB"
        register: result

    - name: Debug the result variable
      debug:
        var: result

    #####
    # Task Name: Create a Journal Volume using required details
    #####
    - name: Create a Journal Volume using required details
      hitachivantara.vspone_block.vsp.hv_journal_volume:
        connection_info: "{{ connection_info }}"
        state: "present"
        spec:
          journal_id: 3
          ldev_ids: [600]

        register: result

    - name: Debug the result variable
      debug:
        var: result
```

Step 5: Create a UR pair

After preparing journal volumes and ensuring remote connectivity, use the `hitachivantara.vspone_block.vsp.hv_hur` module to create UR pairs between primary and secondary volumes.

1. Make sure that the primary volumes (LDEV IDs 500-509) are created and mapped to a host group on the primary storage system.
2. Create a host group, for example, `hur-hg`, on port CL3-A of the secondary storage system before running the playbook. The S-VOLs are automatically created by Ansible in the specified pool and mapped to the host group mentioned in the playbook, in this case, the `hur-hg` host group.
3. Use the `hitachivantara.vspone_block.vsp.hv_hur` module to:
 - a. Create UR pairs with a new copy group.
 - b. Associate both sides with the journal ID.
 - c. Add the pairs to a consistency group.
 - d. Trigger an initial copy of the data.

The playbook loops through each LDEV (500-509), creates a pair, and names each pair dynamically (`ur_copy_pair_500`, `ur_copy_pair_501`, and so on).

The following playbook shows the creation of 10 UR pairs with journal ID 3, with consistency grouping and automatic S-VOL creation:

```
- name: UR Pair Creation Playbook
  hosts: localhost
  gather_facts: false
  vars_files:
    - ../ansible_vault_vars/ansible_vault_storage_var.yml

  vars:

    connection_info:
      address: "{{ storage_address }}"
      username: "{{ vault_storage_username }}"
      password: "{{ vault_storage_secret }}"
    secondary_connection_info:
      address: "{{ secondary_storage_address }}"
      username: "{{ vault_secondary_storage_username }}"
      password: "{{ vault_secondary_storage_secret }}"

  tasks:
    #####
    # Task Name: Create UR pair in new copy group
    #####

    - name: Create UR pairs in new copy group
      hitachivantara.vspone_block.vsp.hv_hur:
        connection_info: "{{ connection_info }}"
        secondary_connection_info: "{{ secondary_connection_info }}"
        state: "present"
        spec:
          copy_group_name: "ur_copy_group_name_1"
          copy_pair_name: "ur_copy_pair_{{ item }}"
          primary_volume_id: "{{ item }}"
          secondary_pool_id: 1
          primary_volume_journal_id: 3
          secondary_volume_journal_id: 3
          local_device_group_name: "ur_local_device_group_name_1"
          remote_device_group_name: "ur_remote_device_group_name_1"
          is_consistency_group: true
          consistency_group_id: 1
          fence_level: "ASYNC"
          secondary_hostgroup:
            name: "hur-hg"
            port: "CL3-A"
          mirror_unit_id: 0
          is_data_reduction_force_copy: false
        loop: "{{ range(500, 510) | list }}"
        register: result

    - name: Debug the result variable
```

```
debug:
  var: result
```

Note:

- To operate on all pairs within a copy group at once, use the `hitachivantara.vspone_block.vsp.hv_remote_copy_group` module instead, which is designed for group-level operations and is more efficient for bulk tasks.
- To view detailed information about a module and all available options, use the `ansible-doc <module name>` command. For example, `ansible-doc hitachivantara.vspone_block.vsp.hv_hur`.

Summary

Deploying Ansible with Universal Replicator streamlines the setup and management of asynchronous replication across storage systems. It automates complex tasks such as journal creation, remote path setup, and volume pairing, reducing configuration time, minimizing human error, and enabling faster, more reliable disaster recovery at scale.

References

- [VSP One Block Storage Modules Ansible User Guide](#)
- [Universal Replicator User Guide](#)
- [UR best practices](#)