



# Disaster Recovery

---

This chapter contains the following topics:

- [Backing Up and Restoring Environment Definitions, on page 1](#)
- [Backup and Restore to the Same Cluster, on page 2](#)
- [Synchronizing MSX Vault and CockroachDB User Passwords, on page 3](#)
- [Restoring to a New Cluster, on page 5](#)
- [Dual Data Center Disaster Recovery, on page 6](#)

## Backing Up and Restoring Environment Definitions

A complete backup of MSX generally includes MSX Data, CockroachDB data, and Environment Definition files. The Environment Definition files contain essential dependencies that are used to run your playbooks. The following files and folders store all the Environment Definition information:

```
/msx-<version>/ansible/group_vars (folder)
/msx-<version>/ansible/inventory/inventory
/msx-<version>/ansible/keys (folder)
/msx-<version>/ansible/ssh.cfg
/msx-<version>/ansible/vms-backup/infra (folder)
/etc/ssl/vms-certs (folder)
```



---

**Note** For details on full backup and restore operations (which include MSX data only), please refer to [Backup and Restoration to Same Cluster](#) and [Restoring to a New Cluster](#).

---

## Backing Up Environment Definitions

Running the `backup-env-defs.yml` playbook will back up the environment definition files (only) and store them in a compressed archive that is encrypted with `ansible-vault`. To perform the backup:

---

**Step 1** Export the `ANSIBLE_VAULT_PASSWORD_FILE` main.yml variable to the path of the password file.

```
export ANSIBLE_VAULT_PASSWORD_FILE=<vault_pwd_path>
```

**Note** If you change the Ansible vault password, then you need to make a new backup. Also, make sure to retain any older Ansible passwords so that you can recover previous backups that used those passwords.

**Step 2** Run the `backup-env-defs.yml` backup playbook in one of two ways:

- a) If you run `backup-env-defs.yml` without a `'backup_tag'` parameter, then the playbook will assign the current date and time stamp for the backup name. For example: `'20200921T152318'`.
- b) If you provide a `<backup_tag>`, then the tag will be used to create the backup name in the following format: `env-def-<backup_tag>.tgz.encrypted.gz`. For example, running `ansible-playbook backup-env-defs.yml --extra-vars="backup_tag=monday-backup"` creates the backup `env-def-monday-backup.tgz.encrypted.gz`.

After you run a backup procedure and the file is successfully copied to S3 (AWS)/MinIO (OpenStack), the script runs an automatic cleanup process to keep the relevant filesystem from getting too full. By default, the script leaves the previous ten backups, but that value can be customized by updating the `"backups_to_keep"` variable in `group_vars/all/main.yml`.

## Restoring Environment Definitions

The `restore-env-defs.yml` playbook is an interactive playbook that allows you to restore environment definition files that were backed up with either the `backup-env-defs.yml` or `vms-backup.yml` playbook.

During the restore operation, a copy of the current environment definitions will be copied to `/tmp/pre-restore-backup/env-def-<backup_tag>`, in case you decide to revert to the existing state for any reason. You will be prompted about overwriting if this directory exists.

Cisco MSX also provides a `"manual_restore"` option that will restore files to `/tmp/restore/env-def-<backup_tag>` and will not move them back to their original directories. You can use this option to restore certain files selectively. You can enable this option by passing `manual_restore=true` into `--extra-vars` when running the playbook. For example:

```
ansible-playbook restore-env-defs.yml --extra-vars="manual_restore=true"
```

There are three ways to run a restore playbook:

1. If your existing system is running, but you do not know the `backup_tag` name, then run `restore-env-defs.yml` to list the existing tags in S3/MinIO and you will be prompted as to which backup you would like to restore. For example: `ansible-playbook restore-env-defs.yml`
2. If you know the `backup_tag` name you would like to restore, then you can specify that as an option. For example: `ansible-playbook restore-env-defs.yml backup_tag=monday-backup`
3. Lastly, if you have the backup file exported from S3/MinIO and placed on the installer container, then you can provide the file path with the `backup_file` option. This option will allow you to restore the file without MSX being deployed. For example: `ansible-playbook test-restore.yml --extra-vars="backup_file=/tmp/env-def-monday-backup.tgz.encrypted.gz"`

## Backup and Restore to the Same Cluster

This procedure is used to restore a data backup to a working MSX cluster where the existing data has been corrupted.

**Step 1** Log in to the installer container.

**Step 2** Navigate to the Ansible directory.

```
cd /msx-4.0.0/ansible
```

**Step 3** Perform a full MSX backup.

```
ansible-playbook vms-backup.yml --extra-var backup_tag=<SPECIFIC_TAG>
```

**Note** Use <SPECIFIC\_TAG> to specify a unique tag that identifies the backup.

All backup files created are stored in the folder /msx-4.0.0/ansible/vms-backup/SPECIFIC\_TAG

**Step 4** Perform a backup of the CockroachDB. For details, see [Backing Up CockroachDB](#).

**Step 5** Copy the vms-backup directory to a place outside the installer container.

In addition to backing up the cluster data by running the above playbook, you should backup all files in the installer container. If the installer container is lost, you can recreate the container and get access to the deployment from the backed up container files. Copy the data under /msx-4.0.0/ansible and /tmp/ansible.log to a location outside the installer container.

**Step 6** Restore the system from an existing backup using the command:

```
ansible-playbook vms-restore.yml --extra-var backup_tag=<SPECIFIC_TAG>
```

**Note** To restore the backup with a <SPECIFIC\_TAG>, pass with that value in extra vars as shown above.

**Step 7** Perform a restore operation for CockroachDB. For details, see [Restoring the CockroachDB](#).

## Synchronizing MSX Vault and CockroachDB User Passwords

After running the `vms-restore.yml` playbook to restore MSX onto a new cluster, the MSX vault and CockroachDB user passwords will be out of sync. As a result, the `ipamservice` will be in a crash loop.

To recover the `ipamservice`:

**Step 1** Log onto the installer container.

**Step 2** Move to the Ansible folder.

```
cd /msx-4.0.0/ansible
```

**Step 3** Export the Ansible Vault password.

```
export ANSIBLE_VAULT_PASSWORD_FILE=<path to file>
```

**Step 4** Run the following to create a playbook that will obtain the restored `ipamservice` secret. Change the <service\_name> to `ipamservice` (or your desired service).

```
cat > get_secret.yml <<EOF
---
- hosts: kube_master[0]
  tasks:
    - name: get client from k8s secret
      run_once: true
```

```

    shell: "kubectl -n {{kubernetes_namespace}} get secret msxvault --no-headers -o
custom-columns=:.data.token | base64 --decode"
    register: client_token
    no_log: "{{ hide_sensitive | default(True) }}"
  - name: set the vault_client_token
    set_fact:
      vault_client_token: "{{ client_token.stdout }}"
    no_log: "{{ hide_sensitive | default(True) }}"
  - name: Retrive secret from vault
    no_log: true
    uri:
      url: "{{ vault_scheme }}://vault.service.consul:8200/v1/secret/{{ service_name }}"
      validate_certs: false
      status_code: 204,200
      headers:
        X-VAULT-TOKEN: "{{ vault_client_token }}"
      method: GET
      body_format: json
    register: password_read
    no_log: "{{ hide_sensitive | default(True) }}"
  - set_fact:
      vault_secret_password: '{{ password_read.json.data["spring.datasource.password"] }}'
  - debug: var=vault_secret_password
EOF

```

**Step 5** Run `get_secret.yml` and set the `service_name` parameter to your desired service name. After running this playbook, the secret will be shown in the output.

```
ansible-playbook get_secret.yml --extra-vars service_name=<service_name>
```

**Step 6** SSH to one of the Kube-Master nodes.

**Step 7** Go to the CockroachDB container.

```
kubectl exec -it cockroachdb-0 -c cockroachdb bash
./cockroach sql --certs-dir cockroach-certs
```

**Step 8** Update the password.

```
ALTER USER <service_name> WITH PASSWORD <service_secret>
```

**Step 9** Verify that the `ipamservice` is running.

**Note** Other MSX services which use CockroachDB will not crash, but their passwords will also be out of sync with the MSX Vault and should be synchronized in the same manner as `ipamservice`. To obtain the passwords for each service, run the `get_secret.yml` playbook with the desired `service_name` as an `extra_vars` parameter.

The other MSX services that use CockroachDB are:

- administrationservice
- billingservice
- catalogservice
- complianceservice
- incidentservice
- notificationservice
- vulnerabilityservice

- workflowservice

---

## Restoring to a New Cluster

Use this procedure in cases where an existing deployment is unusable and unrecoverable. Before performing the restore operation, you must have a backup from a working system that is mounted to your installer container. For more information on the backup procedure, see the [Data Backups](#) section. The overall process will be to create a new installation and then to restore the most recent data from a backup.

- 
- Step 1** If a container does not exist for the restore process, create your MSX installer container. If needed, refer to steps 1-8 in the [Preparing the Container](#) section in the [Upgrade Tasks](#) chapter.
- Step 2** Copy **passwords.yml** from the **vms-backup** directory in the installer container.
- ```
cp /msx-4.0.0/ansible/vms-backup/passwords.yml /msx-4.0.0/ansible/group_vars/all/passwords.yml
```
- Step 3** Copy the certs from either the installer container, the Kube master or, if necessary, from your backup. The cert files are **ca.pem** and **ca-key.pem**, and they are located at **/etc/ssl/vms-certs** on either node. The cert files should be copied onto your installer container into the mounted directory on the host (**/tmp/repo**).
- Step 4** Create your infra node. The new restore cluster should have the same FQDN and AWS region as the backed-up/destroyed cluster. This process will create the target server infrastructure:
- Move to the ansible folder.

```
cd /msx-4.0.0/ansible
```
  - Export the Ansible password file so that you can run Ansible playbooks.

```
export ANSIBLE_VAULT_PASSWORD_FILE=<path to file>
```
  - Build the target-server infrastructure:

```
ansible-playbook create-infra.yml (OpenStack)
ansible-playbook create-infra-aws.yml (AWS)
```
  - Verify that the infrastructure was created properly.

```
ansible-playbook checks/check-createinfra.yml
```
- Step 5** Copy your ca files (**ca.pem** and **ca-key.pem**) to **/etc/ssl/vms-certs** on the installer container.
- Step 6** Proceed with the remainder of the MSX installation. That process will install the necessary services and verify them. For details on the install process, refer to steps 2 and later in the [Installing the MSX Platform in Standalone Mode](#).
- Services being installed:
- Isolated (isolated.yml)
  - Infra Services (deploy-infra-services.yml)
  - Platform Microservices (deploy-vms-microservices.yml)
- Lastly, verify the install:
- ansible-playbook checks/check-vms.yml

**Step 7** Install the Service Packs that were previously installed in the deployment. For details on the process, refer to [Installing Service Packs](#).

**Step 8** Perform the data restore, while making sure that you use the same tag that was specified previously during the backup operation.

```
ansible-playbook vms-restore.yml --extra-vars backup_tag=msx-backup-tag
```

**Note** After running `vms-restore.yml`, `ipamservice` will be in a crash loop. For more information on how to resolve this issue, see [Synchronizing MSX Vault and CockroachDB User Passwords](#).

**Step 9** Perform a restore of the CockroachDB. For details, see [Restoring the Cockroach DB](#).

## Dual Data Center Disaster Recovery

This section contains the following topics:

- [Failing Over to the Passive Data Center](#)
- [Failing Back to the Active Data Center](#)

### Failing Over to the Passive Data Center

Use the following procedure to manually fail over to the passive data center in case the active data center is not reachable or has become unusable in case of a disaster or for other reasons.

**Step 1** Bring down the VPN Tunnel. Log in to the CSR VPN using SSH on the passive data center node and do the following:

```
config t
int tun 0
shut
```

**Step 2** Power off the CSR VPN and CSR HUB VMs on the Active site. Powering off both these VMs will automatically break the tunnel.

**Step 3** Source the OpenStack RC file that was copied to the passive data center node:

```
source vms-backup/infra/openrc-passive
```

**Step 4** Change directory to the `/msx-version/ansible` folder inside the container:

```
cd /msx-4.0.0/ansible
```

**Step 5** Switch to the passive data center:

```
ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: passive}'
```

**Step 6** Stop consul replication on the passive data center:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/consul-replicate-rc.yml" kube-master[0]
```

**Step 7** Switch NSO to become the primary instance on the passive data center:

```
ansible -m command -a "curl -X PUT -d 'Active' -k -H 'X-CONSUL-TOKEN:<consul_root_token>'
https://consul.service.consul:8500/v1/kv/private/dc_status" kube-master[0]
```

The default consul\_root\_token: 73b72724-4570-4000-b720-de18c30fab2

**Step 8** Stop platform microservices on the passive data center:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/administrationservice-rc.yml -f
/etc/kube-manifests/consumeservice-rc.yml -f /etc/kube-manifests/devicemanagerservice-rc.yml -f
/etc/kube-manifests/manageservice-rc.yml -f /etc/kube-manifests/monitorservice-rc.yml -f
/etc/kube-manifests/serviceextensionservice-rc.yml -f /etc/kube-manifests/usermanagementservice-rc.yml
-f /etc/kube-manifests/orchestrationservice-rc.yml -f /etc/kube-manifests/billingservice-rc.yml
-f /etc/kube-manifests/notificationservice-rc.yml" kube-master[0]
```

**Step 9** Start user management on the passive data center and make sure it is up (for orchestration to read ncs streams); and verify if it is up and running:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/usermanagementservice-rc.yml"
kube-master[0]
```

**Step 10** Ensure that the user management service status is up:

```
ansible -m command -a "curl -X GET http://routerservice.service.consul:8765/idm/admin/health"
kube-master[0]
```

**Step 11** Start the platform microservices on the passive data center:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/administrationservice-rc.yml -f
/etc/kube-manifests/consumeservice-rc.yml -f /etc/kube-manifests/devicemanagerservice-rc.yml -f
/etc/kube-manifests/manageservice-rc.yml -f /etc/kube-manifests/monitorservice-rc.yml -f
/etc/kube-manifests/serviceextensionservice-rc.yml -f /etc/kube-manifests/orchestrationservice-rc.yml
-f /etc/kube-manifests/billingservice-rc.yml -f /etc/kube-manifests/notificationservice-rc.yml"
kube-master[0]
```

**Step 12** Restore the AO backup to passive side:

```
ansible-playbook backup-restore-ao.yml --extra-vars "{ BR_mode: restore, backup_tag: (tag from
vms-backup dir) }"
```

**Note** The backup tag `tag from vms-backup dir` will be determined by the admin and is located in the `msx-4.0.0/ansible/vms-backup/` directory.

**Step 13** Stop the data platform probes on the passive data center:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/heartbeat-cm.yml -f
/etc/kube-manifests/sshbeat-cm.yml -f /etc/kube-manifests/snmpbeat-cm.yml -f
/etc/kube-manifests/heartbeat-ps.yml -f /etc/kube-manifests/sshbeat-ps.yml -f
/etc/kube-manifests/snmpbeat-ps.yml" kube-master[0]
```

**Step 14** Start the data platform probes on the passive data center:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/heartbeat-cm.yml -f
/etc/kube-manifests/sshbeat-cm.yml -f /etc/kube-manifests/snmpbeat-cm.yml -f
/etc/kube-manifests/heartbeat-ps.yml -f /etc/kube-manifests/sshbeat-ps.yml -f
/etc/kube-manifests/snmpbeat-ps.yml" kube-master[0]
```

**Step 15** As a superuser, get the IDM token on the passive data center:

```
ansible -m command -a "curl -X POST http://routerservice.service.consul:8765/idm/api/v1/accesstoken
-H 'accept: application/json' -H 'cache-control: no-cache' -H 'content-type: application/json' -d
'{"granttype": "password", "password": "<superuser_password>", "scope": "write",
"username": "superuser"}'" kube-master[0]
```

**Step 16** Push the probe configuration on the passive data center:

```
ansible -m command -a "curl -X POST -H 'authorization: Bearer <token from previous step>'
http://devicemanagerservice.service.consul:9104/devicemanagerservice/v1/admin/devicemetrics/start"
kube-master[0]
```

**Step 17** For the Cisco MSX SD-Branch service pack, run the following playbook:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/nso-vbranch-ps.yml" kube-master[0]
```

**Step 18** For the Cisco MSX Managed Device service pack, run the following playbook:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/nso-manageddevice-ps.yml" kube-master[0]
```

**Step 19** For the SD-Branch service pack, run the following playbook

```
ansible -m command -a "sed -i 's/replicas: 1/replicas: 2/g" kube-master
```

**Step 20** For the Cisco MSX Managed Device service pack, run the following playbook:

```
ansible -m command -a "sed -i 's/replicas: 1/replicas: 2/g" kube-master
```

**Step 21** In the following playbook specify the service packs that have been installed:

```
ansible-playbook deploy-nso-consul-cleanup.yml --extra-vars '{servicepack_list: ['vbranch',
'manageddevice', 'cloudutd']}'
```

**Step 22** Deploy SD-Branch ncs if the SD-Branch service pack is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/nso-vbranch-ps.yml" kube-master[0]
ansible -m command -a "sed -i 's/replicas: 1/replicas: 2/g" kube-master
```

**Step 23** Deploy Managed Device ncs if the Managed Device service pack is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/nso-manageddevice-ps.yml" kube-master[0]
```

**Step 24** Stop the SD-Branch service pack microservice if SD-Branch is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/statemachineservice-rc.yml -f
/etc/kube-manifests/vbranchservice-rc.yml" kube-master[0]
```

**Step 25** Stop the Managed Device service pack microservice if Managed Device is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/manageddeviceservice-rc.yml"
kube-master[0]
```

**Step 26** Stop the SD-WAN service pack microservice if SD-WAN is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/sdwanservice-rc.yml"
kube-master[0]
```

**Step 27** Start the SD-Branch service pack microservice if SD-Branch is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/statemachineservice-rc.yml -f
/etc/kube-manifests/vbranchservice-rc.yml" kube-master[0]
```

**Step 28** Start the Managed Device service pack microservice if Managed Device is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/manageddeviceservice-rc.yml"
kube-master[0]
```

**Step 29** Start the SD-WAN service pack microservice if SD-WAN is installed.

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/sdwanservice-rc.yml" kube-master[0]
```

**Step 30** Run the following playbook to update the DNS entries:

```
ansible-playbook create-infra.yml --tags route53,server
```

- **For SD-Branch:**

Update the PNP hosts of all ENCS devices to point to the Edge Node IP address for the passive data center. Log in to the ENCS NFVIS system, and provide the PNP server IP address or FQDN for the passive data center in the Host Plug-n-Play settings.

**Note** This step is required only when NFVIS version used is lower than 3.8.1 and the IP address is provided for PNP.

If you are using NFVIS 3.8.1, which has FQDN support, the FQDN name can be provided directly (For example: orange-customer.com) and this does not need to be updated after the failover.

- **For SD-WAN:**

After the failover, execute the following steps before proceeding with service operations:

- Update the PNP hosts of all ENCS devices to point to the Edge Node IP address for the passive data center.

**Note** This step is required only when the NFVIS version used is lower than 3.8.1 and the IP address is provided for PNP.

If you are using NFVIS 3.8.1, which has FQDN support, the FQDN name can be provided directly (For example: orange-customer.com) and this does not need to be updated after the failover.

- Verify that the new passive data center NAT IP addresses have been opened out for vOrchestrator connectivity before creating control planes.

- For Managed Device:**

After the failover, perform the following step on NSO through `ncs_cli`:

- Verify if the devices are in sync:

```
request devices check-sync
```

- If the devices are out-of-sync, run this command:

```
request devices sync-from
```

**Note** After the last step is performed, ensure that the ncs for either SD-Branch, or Managed Device, or both, is ready. If ncs is ready, enter this command:

```
ansible -m shell -a "kubectl -n vms get pod | grep nso" kube-master[0]
```

If ncs is ready, then the return output is as follows.

```
nso-manageddevice-0 3/3 Running
nso-vbranch-0 3/3 Running
```

**Step 31** Source the OpenStack RC file that was copied to the passive data center node:

```
vms-backup/infra/openrc-active
```

**Step 32** Switch to active DC using the following command:

```
ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: active}' ----
```

**Step 33** Stop SD-Branch SP microservice if SD-Branch is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/
statemachineservice-rc.yml -f /etc/kube-manifests/vbranchservice-rc.yml"
kube-master[0]
```

**Step 34** Stop Managed Device SP microservice if Managed Device is installed:

```
ansible -m command -a "kubectl delete -f /etc/
kube-manifests/manageddeviceservice-rc.yml" kube-master[0]
```

**Step 35** Scale SD-Branch NSO count to 0 for inactive passive DC if SD-Branch service pack is installed.

```
ansible -m command -a "kubectl scale --replicas=0 -f
/etc/kube-manifests/nso-vbranch-ps.yml" kube-master[0]
```

**Step 36** Scale manageddevice NSO count to 0 for inactive passive DC if manageddevice ServicePack is installed.

```
ansible -m command -a "kubectl scale --replicas=0 -f
/etc/kube-manifests/nso-manageddevice-ps.yml" kube-master[0]
```

**Step 37** Log in to the MSX Portal and create a tenant. For more information, see ‘Logging in to the Portal’ and ‘Managing Tenants’ in the [Cisco Managed Services Accelerator \(MSX\) Platform User Guide](#).

### What to do next

[Enabling the VPN](#).

## Enabling the VPN

After the failover, use the following procedure to enable the VPN and allow the passive data center to send any device configuration and database changes to active data center.

**Step 1** Source the OpenStack RC file that was copied to the active data center node:

```
source vms-backup/infra/openrc-active
```

**Step 2** Switch to the active data center:

```
ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: active}'
```

**Step 3** Switch NSO to become the secondary instance on the active data center:

```
ansible -m command -a "curl -X PUT -d 'Passive' -k -H 'X-CONSUL-TOKEN:<consul_root_token>'
https://consul.service.consul:8500/v1/kv/private/dc_status" kube-master[0]
```

**Step 4** Start consul replication on the passive data center:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/consul-replicate-rc.yml" kube-master[0]
```

**Step 5** Source the OpenStack RC file that was copied to the passive data center node:

```
source vms-backup/infra/openrc-passive
```

**Step 6** Switch to the passive data center:

```
ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: passive}'
```

**Step 7** Bring up the VPN Tunnel. Log in to the CSR VPN using SSH on the active data center node and do the following:

```
Enter into config mode (config t)
int tun 0
start
```

**Note** You can automate the failover steps in AWS and OpenStack using automation scripts. A sample script, `/failover-aws.sh` has been provided under `/msx-4.0.0/ansible/`.

## Failing Back to the Active Data Center

Use the following procedure to manually fail back to the active data center after a fail over.

**Step 1** Bring down the VPN Tunnel. Log in to the CSR VPN using SSH on the active data center node and do the following:

```
config t
int tun 0
shut
```

**Step 2** Power off CSR VPN and CSR HUB VMs on the passive site. Powering off both these VMs will automatically break the tunnel.

**Step 3** Source the OpenStack RC file that was copied to the active data center node:

```
source vms-backup/infra/openrc-active
```

**Step 4** Switch to active data center:

```
ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: active}'
```

**Step 5** Stop consul replication on active data center:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/consul-replicate-rc.yml" kube-master[0]
```

**Step 6** Switch NSO to become the primary instance on the active data center:

```
ansible -m command -a "curl -X PUT -d 'Active' -k -H 'X-CONSUL-TOKEN:<consul_root_token>'
https://consul.service.consul:8500/v1/kv
/private/dc_status" kube-master[0]
```

The default `consul_root_token`: `73b72724-4570-4000-b720-de18c30fabe2`

**Step 7** Stop platform microservices on active data center:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/administrationservice-rc.yml -f
/etc/kube-manifests/consumeservice-rc.yml -f /etc/kube-manifests/devicemanagerservice-rc.yml -f
/etc/kube-manifests/manageservice-rc.yml -f /etc/kube-manifests/monitorservice-rc.yml -f
/etc/kube-manifests/serviceextension-service-rc.yml -f
/etc/kube-manifests/usermanagementservice-rc.yml -f /etc/kube-manifests/orchestrationservice-rc.yml
-f /etc/kube-manifests/billingservice-rc.yml -f /etc/kube-manifests/notificationservice-rc.yml"
kube-master[0]
```

**Step 8** Start user management on active data center, ensure that it's up (for orchestration to read ncs streams); and then verify if it's up and running:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/usermanagementservice-rc.yml"
kube-master[0]
```

**Step 9** Ensure that the user management service status is up:

```
ansible -m command -a "curl -X GET http://routerservice.service.consul:8765/idm/admin/health"
kube-master[0]
```

**Step 10** Start platform microservices on active data center:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/administrationservice-rc.yml -f
/etc/kube-manifests/consumeservice-rc.yml -f /etc/kube-manifests/devicemanagerservice-rc.yml -f
/etc/kube-manifests/manageservice-rc.yml -f /etc/kube-manifests/monitorservice-rc.yml -f
/etc/kube-manifests/serviceextensionservice-rc.yml -f /etc/kube-manifests/orchestrationservice-rc.yml
-f /etc/kube-manifests/billingservice-rc.yml -f /etc/kube-manifests/notificationservice-rc.yml"
kube-master[0]
```

**Step 11** Stop data platform probes on the active data center:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/heartbeat-cm.yml -f
/etc/kube-manifests/sshbeat-cm.yml -f /etc/kubemanifests/
snmpbeat-cm.yml -f /etc/kube-manifests/heartbeat-ps.yml -f /etc/kube-manifests/sshbeat-ps.yml -f
/etc/kube-manifests
/snmpbeat-ps.yml" kube-master[0]
```

**Step 12** Start data platform probes on the active data center:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/heartbeat-cm.yml -f
/etc/kube-manifests/sshbeat-cm.yml -f /etc/kubemanifests/snmpbeat-cm.yml -f
/etc/kube-manifests/heartbeat-ps.yml -f /etc/kube-manifests/sshbeat-ps.yml -f /etc/kube-manifests
/snmpbeat-ps.yml" kube-master[0]
```

**Step 13** As a superuser, get IDM token on the active data center:

```
ansible -m command -a "curl -X POST http://routerservice.service.consul:8765/idm/api/v1/accesstoken
-H 'accept: application/json' -H'cache-control: no-cache' -H 'content-type: application/json' -d
'{"granttype": "password", "password": "<superuser_password>", "
scope": "write", "username": "superuser"}'" kube-master[0]
```

**Step 14** Push the probe configuration on the active data center:

```
ansible -m command -a "curl -X POST -H 'authorization: Bearer <token from previous step>'
http://devicemanagerservice.service.consul:9104/devicemanagerservice/v1/admin/devicemetrics/start
" kube-master[0]
```

**Step 15** For the SD-Branch service pack, run the following playbook:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/nso-vbranch-ps.yml" kube-master[0]
```

**Step 16** For the Managed Device service pack, run the following playbook:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/nso-manageddevice-ps.yml" kube-master[0]
```

**Step 17** Specify only one service pack name if single servicepack has been installed:

```
ansible-playbook
deploy-nso-consul-cleanup.yml --extra-vars '{servicepack_list: ['vbranch', 'manageddevice']}'
```

**Step 18** Deploy SD-Branch ncs if the SD-Branch service pack is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/nso-vbranch-ps.yml"
kube-master[0]
```

**Step 19** Deploy Managed Device ncs if the Managed Device service pack is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/nso-manageddevice-ps.yml" kube-master[0]
```

**Step 20** For the SD-Branch service pack, run the following playbook:

```
ansible -m command -a "sed -i 's/replicas: 2/replicas: 1/g" kube-master
```

**Step 21** For the Managed Device service pack, run the following playbook:

```
ansible -m command -a "sed -i 's/replicas: 2/replicas: 1/g" kube-master
```

**Step 22** Stop the SD-Branch service pack microservice if SD-Branch is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/statemachineservice-rc.yml -f /etc/kube-manifests/vbranchservice-rc.yml" kube-master[0]
```

**Step 23** Stop the Managed Device service pack microservice if the Managed Device is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/manageddeviceservice-rc.yml" kube-master[0]
```

**Step 24** Stop the SD-WAN service pack microservice if SD-WAN is installed:

```
ansible -m command -a "kubectl delete -f /etc/kube-manifests/sdwanservice-rc.yml" kube-master[0]
```

**Step 25** Start the SD-Branch service pack microservice if SD-Branch is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/statemachineservice-rc.yml -f /etc/kube-manifests/vbranchservice-rc.yml" kube-master[0]
```

**Step 26** Start the Managed Device service pack microservice if Managed Device is installed:

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/manageddeviceservice-rc.yml" kube-master[0]
```

**Step 27** Start the SD-WAN service pack microservice if SD-WAN is installed.

```
ansible -m command -a "kubectl create -f /etc/kube-manifests/sdwanservice-rc.yml" kube-master[0]
```

**Step 28** Run the following playbook to update the DNS entries:

```
ansible-playbook create-infra.yml --tags route53,server
```

- **For SD-Branch:**

Update the PNP hosts of all ENCS devices to point to the Edge Node IP address for the active data center. Log in to the ENCS NFVIS system, and provide the PNP server IP address or the FQDN for the active data center in the Host Plug-n-Play settings.

**Note** This step is required only when NFVIS version used is lower than 3.8.1 and IP address is provided for PNP.

If you are using NFVIS 3.8.1, which has FQDN support, the FQDN name can be provided directly (For example: orange-customer.com) and this does not need to be updated after the failover.

- **For SD-WAN:**

After the failover, execute the following steps before proceeding with service operations:

a. Update the PNP hosts of all ENCS devices to point to the Edge Node IP address for the passive data center.

**Note** This step is required only when NFVIS version used is lower than 3.8.1 and IP address is provided for PNP.

If you are using NFVIS 3.8.1, which has FQDN support, the FQDN name can be provided directly (For example: orange-customer.com) and this does not need to be updated after the failover.

Verify the active data center NAT IP addresses have been opened out for vOrchestrator connectivity before creating control planes.

**• For Managed Device**

After the failover, perform the following step on NSO through ncs\_cli:

Verify if the devices are in synch:

```
request devices check-sync
```

If the devices are out-of-sync, use this command:

```
request devices sync-from
```

**Note** After the last step is performed, ensure SD-Branch or managed device ncs, or both are ready. If ncs is ready, enter this command:

```
ansible -m shell -a "kubectl -n vms get pod | grep nso" kube-master[0]
```

If ncs is ready, the return output is as follows.

**Example:**

```
nso-manageddevice-0 3/3 Running
nso-vbranch-0 3/3 Running
```

**Step 29** Log in to the MSX Portal and create a tenant. For more information, see ‘Managing Tenants’ in the [Cisco Managed Services Accelerator \(MSX\) Platform User Guide](#).

**What to do next**

[Enabling the VPN.](#)

**Enabling the VPN**

After failback, use the following procedure to enable the VPN and allow active data center to send any device configuration and database changes to the passive data center.

**Procedure**

|               | Command or Action                                                             | Purpose                                                                                                                                                                                  |
|---------------|-------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Step 1</b> | Source the OpenStack RC file that was copied to the passive data center node: | <code>source vms-backup/infra/openrc-passive</code>                                                                                                                                      |
| <b>Step 2</b> | Switch to the passive data center:                                            | <code>ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: passive}'</code>                                                                                                          |
| <b>Step 3</b> | Switch NSO to become the secondary instance on the passive data center:       | <code>ansible -m command -a "curl -X PUT -d 'Passive' -k -H 'X-CONSUL-TOKEN:&lt;consul_root_token&gt;' https://consul.service.consul:8500/v1/kv/private/dc_status" kube-master[0]</code> |
| <b>Step 4</b> | Start consul replication on the passive data center:                          | <code>ansible -m command -a "kubectl create -f /etc/kube-manifests/consul-replicate-rc.yml" kube-master[0]</code>                                                                        |

|               | Command or Action                                                                                            | Purpose                                                                                                                                                                                                                                                                                         |
|---------------|--------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Step 5</b> | Source the OpenStack RC file that was copied to the active data center node:                                 | <code>source vms-backup/infra/openrc-active</code>                                                                                                                                                                                                                                              |
| <b>Step 6</b> | Switch to active data center.                                                                                | <code>ansible-playbook dualdc-switch-dc.yml --extra-vars '{dc: active}'</code>                                                                                                                                                                                                                  |
| <b>Step 7</b> | Bring up the VPN Tunnel. Log in to the CSR VPN using SSH on the active data center node and do the following | <p>Enter into config mode (<code>config t</code>)</p> <pre>int tun 0 start</pre> <p><b>Note</b> You can automate the failover steps in AWS and OpenStack using automation scripts. A sample script, <code>/failover-aws.sh</code> has been provided under <code>/msx-4.0.0/ansible/</code>.</p> |

