

Ansible Playbook Execution Issues

Ansible is a powerful automation tool, but users may encounter various challenges during its use. This document outlines frequently observed problems and their corresponding solutions.

1. Error: Subscription Registration Connectivity Failure

Description:

System registration fails because nodes cannot reach the subscription service endpoint over HTTPS (port 443). This is usually caused by missing outbound access, DNS/proxy misconfiguration, firewall restrictions, or TLS interception issues.

Symptoms:

- `subscription-manager register` fails on multiple nodes
- Connection errors when accessing the endpoint on **:443**
- DNS resolution for the subscription host fails or resolves incorrectly
- Registration works from some networks but not from the affected nodes

Resolution:

- Verify DNS resolution and basic connectivity to the endpoint from affected nodes
- Check firewall/security group rules allowing outbound HTTPS (443)
- Configure proxy settings if required (`HTTP_PROXY/HTTPS_PROXY/NO_PROXY`)
- Validate system time/certificates (TLS fails if time is wrong)
- Retry registration after network/proxy access is fixed

Code

None

```
curl -vk https://demosat-ha.infra.demo.redhat.com:443/rhsm  
subscription-manager config --list | grep proxy
```

2. Error: HostedCluster Update Failed Due to Missing Configuration

Description:

An update to a HostedCluster resource fails after multiple attempts because required configuration fields are missing, invalid, or incomplete. This prevents the controller from reconciling the resource to the desired state.

Symptoms:

- HostedCluster update retries and eventually fails
- Controller or API logs indicate **missing** or **invalid** configuration
- HostedCluster remains in a **Progressing** or **Degraded** state
- Changes to the HostedCluster spec are not applied

Resolution:

- Review HostedCluster validation and controller logs for specific missing fields
- Verify all required configuration values are defined and valid
- Ensure referenced secrets, config maps, and infrastructure resources exist
- Correct the HostedCluster spec and reapply the update
- Retry once the configuration is complete and consistent

3. Error: Missing Dictionary Attribute

Description:

A task fails because it tries to access a key or attribute (x) that does not exist in a dictionary object in post_infra.yml. This usually happens when expected data is missing, renamed, or conditionally not returned.

Symptoms:

- Error message like “**attribute named x missing from dict object**”
- Failure points to a line accessing `dict.x` or `dict['x']`

- The dictionary is empty or lacks the expected key
- Similar tasks fail when relying on dynamic or conditional data

Resolution:

- Verify the dictionary contains the expected key before accessing it
- Use safe access patterns with `default` or `dict.get()`
- Guard the task with conditions (`when`) or validate inputs early
- Confirm upstream tasks correctly populate the dictionary

4. Error: CloudFormation Stack Creation Failed

Description:

Cloud infrastructure provisioning fails because an AWS CloudFormation stack (for example, `base-infra-####`) cannot be created. This is usually caused by invalid template parameters, missing permissions, service quotas, or dependent AWS resources failing to provision.

Symptoms:

- Stack creation fails with **CREATE_FAILED** or ends in **ROLLBACK_COMPLETE**
- Installer/automation logs show *“Failed to create CloudFormation stack base-infra-####”*
- One or more stack resources report failure reasons (IAM, networking, load balancers)
- Re-running without changes fails in the same stage

Resolution:

- Inspect stack events to identify the **first** failing resource and reason
- Verify required IAM permissions (including `iam:PassRole` where applicable)
- Check AWS service quotas/limits (VPCs, ENIs, EIPs, load balancers, vCPU)
- Validate input parameters (region/AZ, subnet IDs, instance types, naming collisions)

- Delete the failed stack (if needed) and retry after fixing the root cause

Code

None

```
aws cloudformation describe-stack-events --stack-name  
<stack-name>  
aws cloudformation describe-stack-resources --stack-name  
<stack-name>
```

5. Error: API Rate Limit During API Server Shutdown

Description:

An API request to create a Kubernetes object fails because the API server is shutting down and responding with Too Many Requests (HTTP 429). This is usually a transient condition caused by API overload, leader transitions, or control plane instability during restarts/upgrades.

Symptoms:

- Object creation fails with **HTTP 429 Too Many Requests**
- Logs mention the API server is **shutting down** or restarting
- Requests succeed intermittently, then fail again during retries
- Other API operations are slow or temporarily unavailable

Resolution:

- Wait for the API server/control plane to stabilize, then retry
- Add exponential backoff with jitter for create/update requests
- Reduce request rate (batching, lower concurrency) during control plane changes
- Check API server health and events to confirm restarts/upgrades are in progress
- If frequent, investigate underlying control plane instability or resource pressure

6. Error: Gitea Resource Update Failed Due to Missing Configuration

Description:

A Gitea resource update fails after multiple attempts because required configuration fields are missing, incomplete, or invalid. This prevents the controller/operator from applying the update and reconciling the resource to the desired state.

Symptoms:

- Gitea update retries and eventually fails after **x attempts**
- Resource remains **Progressing/Degraded** or does not reflect the requested changes
- Controller/operator logs mention **missing** or **insufficient** information
- Referenced secrets/config maps or values are not found or incomplete

Resolution:

- Inspect the Gitea resource and events to identify which fields are missing
- Verify referenced secrets/config maps exist and contain required keys
- Ensure required spec values (URL, admin/user config, storage, ingress/route) are provided
- Fix the resource definition and re-apply the update
- Retry once the operator reports a healthy/ready state

7. Error: KubeVirt VM Provisioning Failed (VMI Missing)

Description:

A KubeVirt VirtualMachine fails to provision because the corresponding VirtualMachineInstance (VMI) was not created. This usually happens when required volumes/DataVolumes are not ready, storage provisioning fails, or the VM controller cannot start the instance.

Symptoms:

- VM stays in **Provisioning** state and never becomes **Running**
- VM condition shows **VMINotExists** / "VMI does not exist"

- Related PVC/DataVolume is **Pending, Failed**, or missing
- Events mention volume, storage, or scheduling issues

Resolution:

- Verify the VM, VMI, DataVolume, and PVC exist in the correct namespace
- Check DataVolume/PVC status and events; wait until the volume is **Bound/Succeeded**
- Confirm the StorageClass supports the requested access mode/volume mode
- Review CDI and KubeVirt controller logs for provisioning failures
- Retry after storage/scheduling issues are resolved (recreate DV if needed)

Code

None

```
oc -n sandbox-sqvvn-ocp4-cluster get vm,vmi,dv,pvc
oc -n sandbox-sqvvn-ocp4-cluster describe vm control
oc -n cdi logs deploy/cdi-deployment --tail=200
```

8. Error: AWS Reservation Creation Failed

Description:

An attempt to create AWS reservations fails because AWS cannot complete the reservation request. This is commonly caused by insufficient permissions, invalid request parameters, account/region limits, or lack of available capacity for the requested reservation type.

Symptoms:

- Reservation creation request fails immediately or after retries
- AWS API/CLI returns an error indicating reservation creation failed
- Logs mention missing permissions, invalid parameters, or capacity/limit issues
- No new reservation appears in the account after the operation

Resolution:

- Review the AWS error message/details to identify the specific failure reason
- Verify required IAM permissions for purchasing/creating reservations
- Validate request parameters (region, instance family, term, offering class)
- Check account quotas/limits and ensure billing/account requirements are met
- Retry after correcting permissions/parameters or selecting an available option

9. Error: cert-manager Webhook Has No Available Endpoints

Description:

Kubernetes validation fails because the cert-manager-webhook service has no ready endpoints. This usually means the webhook pod is not running, not ready, or cannot be reached, so the API server cannot call it for admission validation.

Symptoms:

- Admission/validation requests fail referencing `cert-manager-webhook`
- Error states the webhook service has **no available endpoints**
- `cert-manager-webhook` pods are **Pending**, **CrashLoopBackOff**, or **NotReady**
- `kubectl/oc get endpoints cert-manager-webhook` shows **<none>**

Resolution:

- Check webhook pod health and events; fix Pending/CrashLoop causes
- Verify the service selector matches the webhook pods' labels
- Ensure the `cert-manager-webhook` deployment has ready replicas
- Confirm network policies/firewalls aren't blocking webhook traffic
- Restart cert-manager components if needed after resolving underlying issues

Code

None

```
kubectl -n cert-manager get deploy,svc,endpoints -l  
app.kubernetes.io/component=webhook  
kubectl -n cert-manager describe deploy cert-manager-webhook  
kubectl -n cert-manager get pods -o wide
```

10. Error: AWS Credential Validation Failed

Description:

An AWS API request fails because the provided credentials cannot be validated. This commonly occurs when access keys are missing, invalid, expired, misconfigured, or lack permission to perform the DescribeInstances operation.

Symptoms:

- AWS API calls fail during **DescribeInstances**
- Errors indicate **invalid**, **expired**, or **unauthorized** credentials
- Ansible/AWS SDK reports authentication or signature validation failures
- Other AWS operations using the same credentials also fail

Resolution:

- Verify AWS credentials are correctly configured (env vars, profile, or IAM role)
- Ensure the access key and secret key are valid and not expired
- Confirm the IAM identity has permission for **ec2:DescribeInstances**
- Prefer IAM roles/instance profiles over static credentials when possible
- Retry after fixing credentials or permissions

11. Error: cert-manager Webhook Has No Available Endpoints

Description:

Kubernetes validation fails because the cert-manager-webhook service has no ready endpoints. This usually means the webhook pod is not running, not ready, or cannot be reached, so the API server cannot call it for admission validation.

Symptoms:

- Admission/validation requests fail referencing `cert-manager-webhook`
- Error states the webhook service has **no available endpoints**
- `cert-manager-webhook` pods are **Pending**, **CrashLoopBackOff**, or **NotReady**
- `kubectl/oc get endpoints cert-manager-webhook` shows **<none>**

Resolution:

- Check webhook pod health and events; fix Pending/CrashLoop causes
- Verify the service selector matches the webhook pods' labels
- Ensure the `cert-manager-webhook` deployment has ready replicas
- Confirm network policies/firewalls aren't blocking webhook traffic
- Restart cert-manager components if needed after resolving underlying issues

Code

None

```
kubectl -n cert-manager get deploy,svc,endpoints -l
app.kubernetes.io/component=webhook
kubectl -n cert-manager describe deploy cert-manager-webhook
kubectl -n cert-manager get pods -o wide
```

12. Error: Undefined AWS Credential Variable

Description:

A task fails because the environment section in `start.yml` references `aws_access_key_id`, but the variable is not defined. This typically happens when credentials are expected from variables but are missing or misconfigured.

Symptoms:

- Error stating '`aws_access_key_id`' is **undefined**

- Playbook fails at a task using the `environment :` field
- AWS-related tasks do not start
- Credentials are not found in variables or inventory

Resolution:

- Define the variable in play vars, inventory, or group/host vars
- Use OS environment variables or an AWS profile instead of hardcoding credentials
- Secure credentials with Ansible Vault if defined in files
- Validate variables before use with assertions
- Retry after providing valid credentials

13. Error: OCM Authentication Token Invalid or Expired

Description:

A connection to the OpenShift Cluster Manager (OCM) API fails because the authorization token is expired, invalid, or missing. This prevents authenticated API requests from succeeding.

Symptoms:

- OCM API requests fail with authentication/authorization errors
- Logs indicate **expired**, **invalid**, or **unauthorized** token
- Operations that depend on OCM (cluster info, registration, management) stop or fail
- Repeated retries do not succeed without re-authentication

Resolution:

- Refresh or re-generate the OCM authorization token
- Verify the token is correctly configured where the tool expects it
- Ensure the token has not expired and has required scopes/permissions
- Re-authenticate and retry the operation after updating the token

14. Error: Remote Host Connection Refused, failed to establish a connection

Description:

A connection attempt to a remote host fails repeatedly because the target is actively refusing connections. This typically means the service is not running, not listening on the expected port, or access is blocked by network or firewall rules.

Symptoms:

- Connection attempts fail with “**connection refused**” after multiple retries
- Tasks abort when trying to reach the remote host
- Manual connection tests to the host and port also fail
- The host is reachable by DNS/IP but not accepting connections

Resolution:

- Verify the remote host is running and the target service is started
- Check firewall rules or security groups to allow the required port
- Confirm the correct hostname/IP and port are configured
- Ensure the service is listening on the expected interface
- Retry after connectivity to the host is restored