Common Issues While Executing Ansible Playbooks

Ansible is a powerful automation tool that simplifies configuration management, application deployment, and infrastructure orchestration. However, even experienced users can encounter issues while executing Ansible playbooks. This article explores common Ansible errors, their causes, and step-by-step troubleshooting methods to resolve them efficiently.

1. Control Plane Startup Timeout

Description:

Cluster creation fails because the control plane (`kube-apiserver`) never becomes ready within the timeout period. Usually caused by etcd health, networking/firewall issues on port 6443, resource limits, or certificate/time problems.

Symptoms:

- Installer logs show: `unable to start the controlplane: timeout waiting for process kube-apiserver to start` or `failed to fetch Cluster`.

Resolution:

- Verify control-plane node health and resources (CPU, memory, disk).
- Check load balancer/DNS/firewall allowing access to `api.<cluster>:6443`.
- Ensure `etcd` is healthy and reachable.
- Confirm certificates are valid and NTP time sync is correct.
- Retry after fixing root causes.

2. ROSA Authentication Token Error

Description:

ROSA login fails because the authorization token used for authentication is expired, invalid, or missing. This commonly occurs when the Red Hat or AWS authentication session has timed out or credentials were rotated.

Symptoms:

- Login command fails with messages indicating an expired or invalid token.
- ROSA or OpenShift CLI commands are rejected immediately during authentication.

Resolution:

- Re-authenticate to refresh the token (Red Hat and/or AWS).
- Verify the correct AWS account, profile, and permissions are in use.
- Ensure the ROSA CLI and OpenShift CLI are up to date.

Code:

```shell
rosa logout
rosa login
aws sts get-caller-identity
rosa whoami
```

3. Job Retrieval Error: Ansible job not found.

Description:

Ansible fails to retrieve the status of an asynchronous task because the job result file cannot be found. This usually happens when the async job ID is checked under a different user or context, the job results were cleaned up, or the job never started correctly.

Symptoms:

- Playbook fails with: `could not find job`
- Error references `ansible_job_id` or a missing results file under `~/.ansible_async/`.

Resolution:

- Ensure `async_status` runs with the same `become`, `remote_user`, and host as the original async task.
- Verify the job ID (`ansible_job_id`) is correct and not overwritten.
- Check that the `~/.ansible_async/` directory exists and is not cleaned up between tasks.
- Avoid switching hosts or using `delegate_to` between async start and status check.

4. AWS Instance Type Unavailable in AZ

Description:

Cluster provisioning fails because the requested EC2 instance type (`i3.metal`) is not available in the selected availability zone (`us-east-2a`). This can happen due to AZ capacity limits, instance-type restrictions, or because the instance type isn't offered in that AZ for your account.

Symptoms:

- Repeated retries, then failure stating the machine type is not found/available in the AZ.
- AWS/installer logs mention `InstanceType` unavailable, `InsufficientInstanceCapacity`, or "not supported in availability zone".

Resolution:

- Try a different availability zone (e.g., `us-east-2b` / `us-east-2c`).
- Choose a different instance type (same family or a modern equivalent).
- If using Spot, switch to On-Demand or broaden allowed instance types.
- Verify account/service limits and any region/AZ restrictions.

Code:

Shell
```shell
aws ec2 describe-instance-type-offerings \
  --location-type availability-zone \
  --filters Name=instance-type,Values=i3.metal \
  --region us-east-2 \
  --query "InstanceTypeOfferings[].Location" \
  --output text
```

5. OpenShift API Connection Refused

Description:

Ansible cannot reach the OpenShift API server because the endpoint is refusing connections (port 6443 not reachable or the API server isn't listening). This is typically caused by DNS/LB misrouting, firewall/security group rules, or an unhealthy control plane.

Symptoms:

- Errors like `Connection refused`, `Max retries exceeded`, or `Failed to establish a new connection` to `https://api.<cluster>:6443/...`
- `oc`/`kubectl` commands to the API endpoint fail from the same host.

Resolution:

- Verify DNS for `api.<cluster>` points to the correct IP/load balancer.
- Ensure port 6443/TCP is open between the Ansible runner and the API endpoint (firewall/SG/LB).
- Check load balancer target health and backend mapping to control-plane nodes.
- Confirm the control plane is up (kube-apiserver running/healthy), then retry.

6. OpenShift Cluster Install Exit Code 4

Description:

Cluster creation failed because `openshift-install create cluster` exited with code **4**, indicating the installer hit an unrecoverable error during provisioning (often API/control-plane bring-up, infrastructure creation, DNS/LB, or permissions).

Symptoms:

- Installer output ends with: `openshift-install ... exited with rc=4` / `non-zero return code 4`
- Logs show earlier failures (e.g., API `:6443` unreachable, `kube-apiserver` timeout, CloudFormation/IAM errors).

Resolution:

- Inspect the installer logs to find the **first root cause** (don't focus on the exit code itself).
- Verify infrastructure prerequisites: DNS, load balancer/API endpoint, networking, IAM permissions/quota.
- Check control-plane status if resources were created (API reachability, `kube-apiserver`, `etcd`).

- Fix the underlying cause, clean up failed resources if needed, and re-run the install.

Code:

```
Shell
openshift-install --dir <install-dir> wait-for install-complete
--log-level=debug
tail -n 200 <install-dir>/.openshift_install.log
```

7. AAD Graph API Access Blocked

Description:

The application cannot access the Azure Active Directory (AAD) **Graph API** because access is blocked by tenant policy, missing permissions/consent, conditional access restrictions, or because the AAD Graph API is disabled/deprecated in your environment.

Symptoms:

- Requests to AAD Graph endpoints fail with `AccessDenied`, `Unauthorized`, or “access blocked” errors.
- Azure logs show the app/service principal lacks required permissions or admin consent.

Resolution:

- Confirm which API the app should use; prefer **Microsoft Graph** instead of AAD Graph when possible.
- Verify the app registration has the required API permissions and that **admin consent** was granted.
- Check Conditional Access / tenant restrictions that may block the app or client type.
- Validate the correct tenant, client ID/secret/cert, and token audience/scope are being used.

Code:

```
Shell
# Verify token works against Microsoft Graph (preferred)
curl -H "Authorization: Bearer $ACCESS_TOKEN"
https://graph.microsoft.com/v1.0/me
```

8. ClusterPolicy Gather Timeout

Description:

Ansible timed out while waiting to retrieve `ClusterPolicy` information (often from an Operator-managed API like NVIDIA GPU Operator). This usually indicates the API/resource is slow or unavailable, the operator isn't ready, RBAC prevents access, or the cluster is under heavy load.

Symptoms:

- Task fails after ~600s with a timeout while fetching `ClusterPolicy`.
- `oc get clusterpolicy` hangs, returns nothing, or shows resources stuck in `Pending/NotReady`.
- Operator pods are not Ready or are crashlooping.

Resolution:

- Check the operator and its CR status (pods, events, conditions) and ensure it's fully installed/Ready.
- Verify RBAC: the service account running Ansible can `get/list/watch` `clusterpolicies`.
- Confirm the API group is available and responsive (`oc api-resources | grep -i clusterpolicy`).
- Increase the timeout/retries only after confirming the operator is progressing normally.
- Investigate cluster pressure (CPU/memory/API server latency) if requests are slow.

9. OpenShift Pipelines Operator Installation Failed

Description:

Ansible failed to install the `openshift-pipelines-operator` after multiple attempts. This usually happens when OperatorHub/catalog sources are unavailable, the subscription/CSV is stuck, required namespaces or OperatorGroup are missing/misconfigured, or the cluster cannot pull images.

Symptoms:

- Subscription exists but never reaches `AtLatestKnown` / CSV stays `Pending` or `InstallReady` fails.
- `InstallPlan` stuck or failing; Operator pod not created or crashlooping.
- Events mention catalog source errors, image pull failures, or permission/namespace issues.

Resolution:

- Verify OperatorHub and CatalogSources are healthy and reachable (network/proxy/air-gapped considerations).
- Check the `Subscription`, `InstallPlan`, and `CSV` status and events to find the first failure reason.
- Confirm the target namespace exists and has a valid `OperatorGroup`.
- Ensure image pulls work (registry access, pull secrets, proxy config).
- If stuck, delete/recreate the Subscription (or approve InstallPlan if manual approval is enabled), then retry.

Code:

Shell
```shell
oc get catalogsources -n openshift-marketplace
oc get sub,ip,csv -n openshift-operators
oc describe sub openshift-pipelines-operator -n openshift-operators
oc describe csv -n openshift-operators | sed -n '1,160p'
```

10. Cluster Infrastructure Creation Failure (openshift-install RC=4)

Description:

Cluster creation fails because `openshift-install` exits with a non-zero return code (RC=4) while creating infrastructure resources. This typically indicates a problem with cloud resources, permissions, quotas, or invalid configuration during the infrastructure provisioning phase.

Symptoms:

- `openshift-install` exits with return code **RC=4**
- Logs show errors while creating infrastructure resources (VPC, IAM roles, load balancers)
- Cloud provider reports stack or resource creation failures

Resolution:

- Review installer logs to find the first failing infrastructure resource
- Verify cloud credentials and required permissions (IAM roles/policies)
- Check service quotas and limits (networks, IPs, load balancers)
- Validate `install-config.yaml` parameters (region, instance types, networking)
- Delete failed resources/stacks and re-run the installer

Code:

Shell
```shell
openshift-install create cluster --log-level=debug
aws cloudformation describe-stack-events --stack-name
<cluster-stack-name>
```

11. Remote Connection Refused

Description:

Ansible fails to connect to a remote host because the connection is actively refused. This usually means the target service is not running, not listening on the expected port, or is blocked by network or firewall rules.

Symptoms:

- Connection attempts fail with **“connection refused”**

- Tasks time out or stop immediately when reaching the remote host
- Manual connection attempts to the host and port also fail
- The target host is reachable by DNS/IP but not accepting connections

Resolution:

- Verify the target host is running and the expected service is listening
- Check firewall rules or security groups to ensure the port is allowed
- Confirm the correct hostname/IP and port are configured
- Ensure the service on the target host has started successfully
- Retry the task after restoring connectivity

Code:

```shell
nc -vz <host> <port>
systemctl status <service>
```

12. StorageCluster Information Retrieval Timeout

Description:

Ansible fails to retrieve StorageCluster information within the expected time limit. This usually occurs when storage operators are still initializing, the cluster is under load, or the StorageCluster is not yet ready or reachable.

Symptoms:

- Operation fails with a **timeout** while querying StorageCluster resources
- Logs show messages like *“Failed to gather StorageCluster information after X seconds”*
- Storage-related playbooks or checks hang and then abort
- StorageCluster status remains **Progressing** or **NotReady**

Resolution:

- Verify the StorageCluster and related operators are installed and running
- Check the StorageCluster status and events for delays or failures
- Ensure sufficient cluster resources (CPU, memory, storage) are available
- Increase the timeout value if initialization is expected to take longer
- Retry after the StorageCluster reaches a **Ready** state

Code:

```shell
oc get storagecluster
oc describe storagecluster <name>
oc get pods -n openshift-storage
```

13. Operator Installation Failure Due to Insufficient Capacity

Description:

The cluster fails to install the **amq-streams** operator after multiple attempts because there are not enough available resources to schedule the required operator pods. This typically happens when the cluster lacks sufficient CPU, memory, or node capacity.

Symptoms:

- Operator installation retries and eventually fails
- Logs indicate **insufficient cluster capacity** or unschedulable pods
- Operator pods remain in **Pending** state
- Node events show resource pressure or scheduling failures

Resolution:

- Check cluster node capacity and current resource usage
- Add or scale up nodes to provide additional CPU and memory
- Reduce resource consumption of existing workloads if possible

- Verify operator resource requests and limits are reasonable
- Retry operator installation after capacity issues are resolved

Code:

Shell

```shell
oc get pods -n openshift-operators
oc describe pod <operator-pod>
oc get nodes
```