**DevOps Case Study: Implementing GitOps Workflow using Flux and Kubernetes**

**1. Cluster Setup**

**Cause 1**

**Error:**

Unable to connect to the server: dial tcp <IP>:443: i/o timeout

A screenshot of a computer screen

AI-generated content may be incorrect.

**Root Cause:**  
Terraform successfully provisioned the EKS cluster, but kubectl was unable to access it due to a missing or misconfigured kubeconfig.

**Resolution:**  
Ensure the kubeconfig file is correctly exported and that the context is set to the right cluster:

aws eks --region eu-central-1 update-kubeconfig --name compredict-eks-cluster

kubectl get nodes

**Cause 2**

**Error:**

EKS Control Plane is not Ready

**Root Causes:**

1. EKS cluster creation still in progress or failed.
2. Missing IAM roles or insufficient permissions (e.g., eks:DescribeCluster, ec2:CreateSecurityGroup).
3. VPC or networking misconfiguration blocking control plane communication.
4. aws-auth ConfigMap not applied, preventing node registration.
5. Incorrect region or cluster name in kubeconfig.

**Troubleshooting Steps:**

1. Check cluster status:
2. aws eks describe-cluster --name compredict-eks-cluster --region eu-central-1 --query "cluster.status"

Wait until the output shows "ACTIVE".

1. Verify IAM roles and attach the following if missing:
   * AmazonEKSClusterPolicy
   * AmazonEKSServicePolicy
   * AmazonEKSWorkerNodePolicy
   * AmazonEKS\_CNI\_Policy
   * AmazonEC2ContainerRegistryReadOnly
2. Confirm VPC configuration and ensure:
   * Private subnets with outbound Internet access.
   * Security groups allow:
     + Inbound on port 443 from worker node SG.
     + Outbound to node subnets.
3. Update kubeconfig and validate connectivity:
4. aws eks update-kubeconfig --name compredict-eks-cluster --region eu-central-1
5. kubectl get svc
6. Apply the aws-auth ConfigMap to allow worker nodes to join:
7. kubectl apply -f aws-auth-configmap.yaml

**2. Node Group Creation Failure**

**Error:**

AsgInstanceLaunchFailures: I’ve reached my aws free tair quota for maximum Fleet Requests for this account. Launching EC2 instance failed.  
  
A screenshot of a computer program

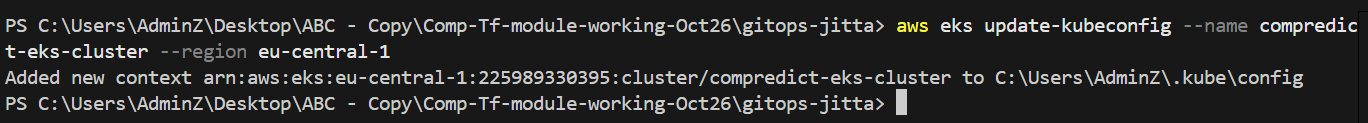
AI-generated content may be incorrect.

**Root Cause:**  
The EKS Node Group creation failed because the AWS free-tier account reached the EC2 Fleet Request quota limit.

**Resolution:**

* Request a **quota increase** from AWS or use a **smaller instance type**.
* Reduce desired node count temporarily.
* After adjusting, re-run:
* terraform apply

**Outcome:**  
The EKS cluster and node group were successfully created.  
  
  
  
  
  
Cluster connectivity verified with:

aws eks update-kubeconfig --name compredict-eks-cluster --region eu-central-1  


kubectl get svc  
  
A screenshot of a computer

AI-generated content may be incorrect.

**3. Kubernetes Validation**

**Command:**

kubectl get svc  
  
A screenshot of a computer

AI-generated content may be incorrect.

**Purpose:**  
Lists all services running in the cluster within the default namespace.  
At this stage, only the default kubernetes service is visible, confirming that the cluster API server is active.

**Next Steps:**  
To verify workloads and deployments:

kubectl get pods -A  
A screenshot of a computer

AI-generated content may be incorrect.

kubectl get svc -A  
  
A screenshot of a computer program

AI-generated content may be incorrect.

This lists all pods and services across namespaces.

**4. Flux GitOps Reconciliation**Check if Flux Git source is synced  
The Git repository name (flux-system by default).

The latest fetched revision (commit SHA).  
Whether the source is Ready, Stalled, or Failed.

flux get sources git  
A black background with white text

AI-generated content may be incorrect.

If the status is Ready → Flux successfully pulled the latest Git changes.  
If it shows Stalled or Failed → Run reconciliation manually:

Check Kustomization synchronization  
flux get kustomizations

**Command:  
  
flux reconcile source git flux-system  
A screenshot of a computer screen

AI-generated content may be incorrect.**

**Purpose:**  
Forces Flux to immediately pull the latest configurations from the Git repository instead of waiting for the scheduled sync.

**Possible Root Causes:**

* Git authentication failure (missing or invalid SSH key/token).
* Incorrect branch or path configuration in the GitRepository manifest.
* Network or DNS connectivity issues preventing Flux from reaching the Git provider.
* Repository not updated (commit hash unchanged).

**Resolution:**  
Cross-verify Git credentials, repository configuration, and network access.  
Re-run the command once verified:

flux reconcile source git flux-system

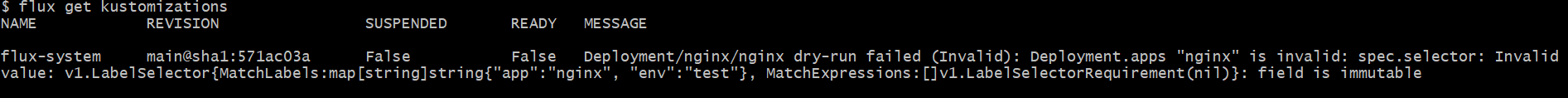
**Command:**

flux reconcile kustomization flux-system  
A screenshot of a computer program

AI-generated content may be incorrect.

**Purpose:**  
Triggers Flux to apply the latest Kubernetes manifests defined in the flux-system Kustomization.

**Verification Commands:**

flux get kustomizations  
  


flux logs -n flux-system --level=error  
A black background with many small white and blue lines

AI-generated content may be incorrect.

**Resolution:**

* Validate all YAML manifests for syntax correctness.
* Ensure all dependent resources (namespaces, CRDs) exist.
* Check and fix any RBAC permission issues.
* Re-run reconciliation to apply the manifests successfully.

**Final Results.**

Successfully provisioned an **EKS cluster using Terraform**.  
 Integrated **FluxCD for GitOps-based continuous delivery**.  
Validated cluster health, node group readiness, and Git repository synchronization.  
Implemented best practices for IAM, networking, and GitOps automation.