**V3-TESTS**

**1-NODE DEPLOYMENT AND CONFIGURATION**

**Test Name:**

New node group deployment on isv-cluster using Crossplane claim file

**Description:**

This test validates the ability to deploy a new node group in a isv-cluster using Crossplane claim files. It ensures that the claim file is processed correctly, resources are provisioned in AWS, and the isv-cluster updates its node group configuration successfully.

**Setup:**

1. Tools and Dependencies:

   - Ensure Crossplane is installed and operational on the regional cluster.

   - ArgoCD is configured on the regional cluster to manage Crossplane claim files.

2. Environment:

   - An active regional cluster with Crossplane installed.

   - A pre-configured GitOps repository managed by ArgoCD for isv-cluster claims.

3. Resources:

   - A Crossplane claim file defining the node group configuration.

**Procedure:**

1. Pre-Test Validation:

   - Confirm that the isv-cluster is operational and connected:

    -   kubectl get nodes --context=<isv-cluster-context>

   - Check ArgoCD application sync status for the isv-cluster claims repository:

    -   argocd app get <isv-cluster-claim-app>

2. Create Crossplane Claim File:

   - Add a new claim file 'claim.yaml' in the GitLab repository managed by ArgoCD

   - Commit and push the file to the Git repository.

3. Monitor ArgoCD for Syncing the Claim:

   - Check the ArgoCD application status to ensure synchronization

   - Confirm the claim is applied to the regional cluster

4. Verify Node Group Creation in AWS:

   - Check AWS Management Console or use AWS CLI to confirm node group creation

5. Verify Node Group Status in isv-cluster:

   - Confirm new nodes are added to the isv-cluster

**Results:**

- Expected Outcome:

  - The claim file triggers the creation of a new node group in AWS.

  - The isv-cluster registers the new nodes and scales up as per the specified configuration.

- Pass Criteria:

  - The node group is visible in AWS EKS.

  - New nodes are added to the isv-cluster and are in a `Ready` state.

**Observation:**

**Test Name:**

Node Auto-Scaling on isv-cluster Using Crossplane Claim File

**Description:**

This test validates the auto-scaling functionality of a node group within a isv-cluster managed by Crossplane. It ensures that the scaling configuration specified in the claim file is applied and that the cluster scales up or down based on workload demand.

**Setup:**

1. Tools and Dependencies:

   - Crossplane installed and operational on the regional cluster.

   - ArgoCD configured to sync Crossplane claim files for isv-clusters.

2. Environment:

   - An active isv-cluster with node groups created and managed via Crossplane.

   - A workload deployed to the isv-cluster capable of generating CPU or memory demand.

3. Resources:

   - A Crossplane claim file with auto-scaling configuration for the node group.

**Procedure:**

1. Pre-Test Validation:

   - Verify that the isv-cluster and node groups are operational

   - Ensure the auto-scaling configuration is defined in the claim file

2. Deploy the Workload:

   - Deploy a workload that can generate resource demand, if necessary

3. Apply the Claim File:

   - Push the updated claim file to the Git repository managed by ArgoCD

   - Monitor the ArgoCD sync status

4. Monitor Scaling Behavior:

   - Watch the node group in AWS for changes in desired, minimum, or maximum size

   - Check the number of nodes in the cluster as the workload increases

5. Simulate Reduced Demand:

   - Reduce the workload by scaling down the deployment.

   - Verify that the node group scales down after demand decreases.

**Results:**

- Expected Outcome:

  - The node group scales up when workload demand increases and scales down when demand decreases.

  - AWS reflects the changes in node group size as specified by the scaling configuration.

- Pass Criteria:

  - New nodes are added or removed as expected during the test.

  - Cluster remains stable during scaling operations, with no resource contention or node failures.

**Observation:**

**Test Name:**

Node Configuration Validation on Isv-Cluster Using Crossplane Claim File

**Description:**

This test validates that the node group created on the Isv-Cluster using a Crossplane claim file adheres to the specified configuration. The test ensures that instance types, subnets etc. are correctly applied and functional.

**Setup:**

1. Tools and Dependencies:

   - Crossplane installed and operational on the regional cluster.

   - ArgoCD managing Crossplane claim files for Isv-Clusters.

2. Environment:

   - A regional cluster managing Isv-Clusters via Crossplane.

   - A Isv-Cluster with pre-existing configurations and permissions to validate against.

3. Resources:

   - A Crossplane claim file defining the node configuration for the Isv-Cluster.

**Procedure:**

1. Pre-Test Validation:

   - Ensure the Isv-Cluster is active and operational:

   - Validate existing node group configurations in AWS:

2. Deploy the Node Group Claim File:

   - Update or create a Crossplane claim file (`node-cliam.yaml`) specifying the desired node configuration:

   - Commit and push the file to the Git repository managed by ArgoCD.

3. Sync ArgoCD Application:

   - Monitor the sync status of the claim file:

   - Confirm the claim file is applied to the Isv-Cluster

4. Validate Node Configuration in AWS:

   - Check the node group configuration in AWS:

   - Confirm that the instance types, scaling policies, and subnets etc. match the claim file.

5. Validate Nodes in the Isv-Cluster:

   - Check the node instances added to the cluster:

   - Verify that the instance types match the configuration specified in the claim file.

**Results:**

- Expected Outcome:

  - Node group configurations (instance types, scaling policies, subnets, etc) match the claim file.

  - Nodes are registered with the Isv-Cluster and are functional.

  - Workloads are successfully scheduled and executed on the nodes.

- Pass Criteria:

  - AWS reflects the correct node group configuration as defined in the claim file.

  - Isv-Cluster nodes are healthy and match the instance types and subnets specified.

**Observation:**

**Test Name:**

Network Configuration Validation on Isv-Cluster

**Description:**

This test validates the network configuration of a Isv-Cluster created using Crossplane. It ensures that the subnets, VPC, and associated networking resources (security groups, route tables) are configured correctly and align with the specifications in the Crossplane claim file.

**Setup:**

1. Tools and Dependencies:

   - Crossplane installed on the regional cluster.

   - ArgoCD managing the Isv-Cluster's Crossplane claim files.

2. Environment:

   - An operational Isv-Cluster managed by Crossplane with a defined network configuration.

3. Resources:

   - Crossplane claim file specifying the Isv-Cluster network configuration, including:

     - VPC ID

     - Subnets

     - Security groups

   - Test workloads to validate network connectivity.

**Procedure:**

1. Pre-Test Validation:

   - Check the Isv-Cluster status

   - Validate the presence of networking CRDs in Crossplane

2. Deploy Networking Configuration Claim File:

   - Update or create a Crossplane claim file.

   - Commit and push the claim file to the Git repository managed by ArgoCD.

3. Sync ArgoCD Application:

   - Monitor the sync status:

   - Confirm that networking resources are provisioned in AWS:

4. Validate Networking Resources in AWS:

   - Verify the VPC, subnets

   - Check security groups and their rules:

5. Test Connectivity Between Nodes:

   - SSH into one of the Isv-Cluster nodes and test connectivity to other nodes using their private IPs:

**Results:**

- Expected Outcome:

  - VPC, subnets, security groups, and routing configurations match the claim file.

  - Nodes in the Isv-Cluster communicate successfully within the configured subnets.

- Pass Criteria:

  - AWS networking resources are correctly configured and visible.

  - No errors in Crossplane reconciliation logs.

  - Successful network connectivity tests between nodes and workloads.

**Observation:**

**Test Name:**

New AMI ID Validation on ISV-Cluster Using Crossplane Claim File

**Description:**

This test ensures that a new AMI ID specified in a Crossplane claim file for a node group is applied correctly to the ISV-Cluster. It validates that the nodes in the ISV-Cluster use the specified AMI ID and confirms the compatibility of the AMI with the cluster setup and workloads.

**Setup:**

1. Tools and Dependencies:

   - Crossplane installed and operational on the regional cluster.

   - ArgoCD managing Crossplane claim files for ISV-Clusters.

2. Environment:

   - An operational ISV-Cluster managed by Crossplane.

   - Node group created using a Crossplane claim file.

   - GitOps repository for managing claim files via ArgoCD.

3. Resources:

   - A Crossplane claim file specifying the AMI ID for the node group.

**Procedure:**

1. Pre-Test Validation:

   - Verify the ISV-Cluster is active and nodes are operational

   - Ensure the new AMI ID is available and compatible with the cluster

2. Update the Crossplane Claim File:

   - Modify the Crossplane claim file to include the new AMI ID (ng-claim.yaml).

   - Commit and push the claim file to the Git repository managed by ArgoCD.

3. Sync the Claim File in ArgoCD:

   - Monitor the sync status of the claim file.

   - Confirm that the claim is applied.

4. Validate Node Group Configuration in AWS:

   - Check that the new AMI ID is applied to the node group.

   - Confirm that the `launchTemplate` reflects the updated AMI ID.

5. Validate Nodes in the ISV-Cluster:

   - Check that new nodes are added using the specified AMI ID.

   - Verify the node image details in AWS EC2 instances.

**Results:**

- Expected Outcome:

  - The new AMI ID is applied to the node group in AWS.

  - Nodes in the ISV-Cluster are provisioned using the specified AMI.

- Pass Criteria:

  - AWS reflects the updated AMI ID in the node group configuration.

  - Nodes are healthy and operational with the new AMI.

**Observation:**

**Test Name:**

Testing High-Availability on ISV-Cluster

**Description:**

This test validates the high-availability (HA) setup of a ISV-Cluster managed through Crossplane. It ensures that the ISV-Cluster can handle node failures, continues to function under stress, and maintains service availability by leveraging multiple availability zones and node groups.

**Setup:**

1. Tools and Dependencies:

   - Crossplane installed and operational on the regional cluster.

   - ArgoCD managing Crossplane claim files for ISV-Clusters.

2. Environment:

   - A ISV-Cluster provisioned with nodes distributed across multiple availability zones (AZs).

   - Workloads deployed with appropriate pod anti-affinity rules for HA.

3. Resources:

   - A Crossplane claim file specifying a multi-AZ setup for the node group.

**Procedure:**

1. Pre-Test Validation:

   - Ensure the ISV-Cluster is operational and nodes are distributed across multiple AZs.

   - Verify the node group configuration in AWS.

2. Deploy High-Availability Workload:

   - Deploy a sample workload with at least three replicas and a load balancer.

   - Apply the workload.

3. Simulate Node Failures:

   - Identify nodes hosting the workload pods.

   - Terminate one of the nodes in AWS.

   - Monitor pod rescheduling and node recovery.

4. Simulate Increased Load:

   - Scale the workload to simulate increased traffic.

   - Monitor node autoscaling behavior and ensure load distribution.

**Results:**

- Expected Outcome:

  - The workload remains available despite node failures.

  - Pods are rescheduled automatically to other nodes in different AZs.

  - Autoscaling adds nodes to handle increased traffic.

- Pass Criteria:

  - No service downtime is observed during node failure or recovery.

  - Workload replicas are evenly distributed across AZs.

**Observation:**

**Test Name:**

NodeGroup Rollout on ISV-Cluster

**Description:**

This test validates the rollout of a new or updated NodeGroup in a ISV-Cluster managed using Crossplane. It ensures the correct provisioning of nodes, adherence to the specified configuration, and minimal disruption to workloads during the rollout process.

**Setup:**

1. Tools and Dependencies:

   - Crossplane installed on the regional cluster.

   - ArgoCD managing Crossplane claim files for ISV-Clusters.

2. Environment:

   - A functioning ISV-Cluster with at least one active NodeGroup.

   - Workloads deployed on the ISV-Cluster to observe impact during the rollout.

3. Resources:

   - A Crossplane claim file defining the new or updated NodeGroup.

**Procedure:**

1. Pre-Test Validation:

   - Verify the ISV-Cluster is active.

   - Confirm the existing NodeGroup configuration in AWS.

2. Prepare the NodeGroup Rollout Claim File:

   - Create or update the Crossplane claim file to define the new NodeGroup or update an existing one.

   - Commit and push the file to the Git repository managed by ArgoCD.

3. Sync the Claim File in ArgoCD:

   - Monitor the sync process to ensure the NodeGroup claim is applied.

   - Confirm the NodeGroup is being created or updated.

4. Validate NodeGroup Rollout in AWS:

   - Verify the new or updated NodeGroup configuration in AWS:

   - Confirm the desired configuration.

5. Test Node Availability in ISV-Cluster:

   - Check that nodes from the new NodeGroup join the cluster:

   - Confirm that existing workloads are rescheduled to the new nodes if necessary.

6. Deploy Workloads on the Updated NodeGroup:

   - Deploy a test workload to validate functionality.

   - Apply the workload.

   - Confirm the workload is running on the new NodeGroup.

**Results:**

- Expected Outcome:

  - The new NodeGroup is successfully created or updated in AWS and joins the ISV-Cluster.

  - Existing workloads remain unaffected, and new workloads are scheduled on the updated NodeGroup.

  - Configuration matches the specifications in the claim file.

- Pass Criteria:

  - AWS and Kubernetes reflect the updated NodeGroup configuration.

  - New nodes join the cluster without errors.

  - Workloads are successfully scheduled and operate on the updated NodeGroup.

**Observation:**