EKS Worker Node Termination and Recreation Process as Part of the EKS Cluster Worker Node Disaster Recovery Plan

To simulate a worker node termination and ensure that Karpenter correctly creates a new node as part of a disaster recovery drill in EKS, follow these steps:

Pre-requisites:

1. **EKS Cluster**: Ensure you have an existing EKS cluster.
2. **Karpenter**: Make sure Karpenter is properly configured and running in your cluster.
3. **AWS CLI**: Install and configure the AWS CLI with appropriate permissions.
4. **kubectl**: Ensure kubectl is configured to interact with your EKS cluster.

Steps:

- **Identify the Worker Node to Terminate:**
  - Use `kubectl` to list all the nodes in your cluster and identify the node you want to terminate.

```
kubectl get nodes
```

- **Drain the Node (Optional but Recommended):**
  - Draining a node will safely evict all the pods running on it.

```
kubectl drain <node-name> --ignore-daemonsets --delete-local-data
```

- **Terminate the Node:**
  - Use the AWS Management Console or AWS CLI to terminate the identified EC2 instance.
  - Using AWS CLI:

```
aws ec2 terminate-instances --instance-ids <instance-id>
```

  - You can find the instance ID of the node by describing the node in kubectl:

```
kubectl describe node <node-name> | grep "ProviderID"
```

The `ProviderID` will give you the instance ID.

- **Observe Karpenter Creating a New Node:**
  - Karpenter should detect the need for a new node and start provisioning one automatically.
  - You can monitor the Karpenter logs to see its actions:

```
kubectl logs -f -n karpenter <karpenter-pod-name>
```

- **Verify New Node Creation:**
  - After a few minutes, check if a new node has been created and joined the cluster.

```
kubectl get nodes
```

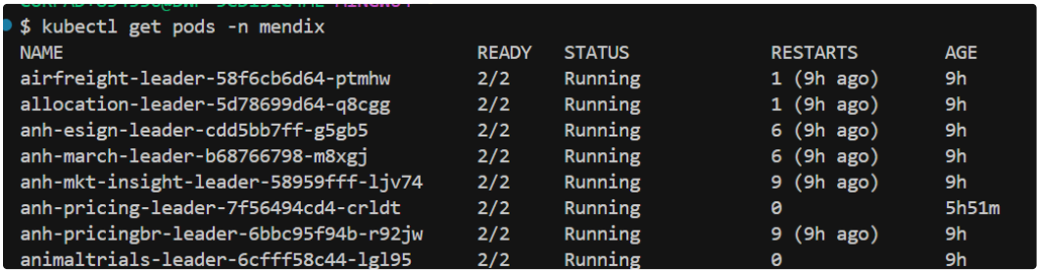
  - Ensure that the new node is in a `Ready` state and that pods are being scheduled on it.
- **Validate the Setup:**
  - Ensure that the pods terminated on the old node are created in the new node and the pods are up and running properly by using below commands
  -
```
kubectl get pods -n mendix
```

```
kubectl logs <pod name> -n mendix
```

  - Verify that all critical applications and services are running as expected.
- **In case pod state is not in `Ready` state**

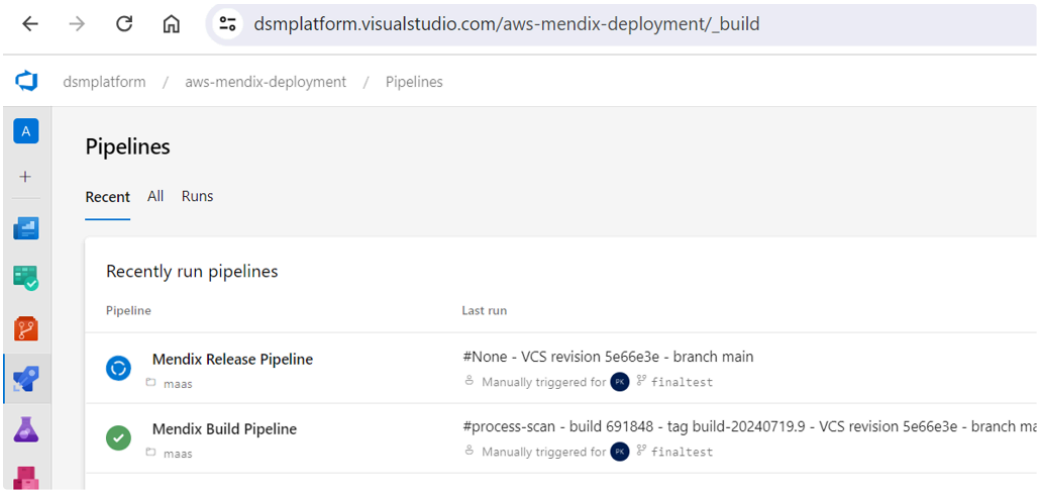
Please check the pod logs and take action as needed.

```
kubectl logs <pod name> -n mendix
```

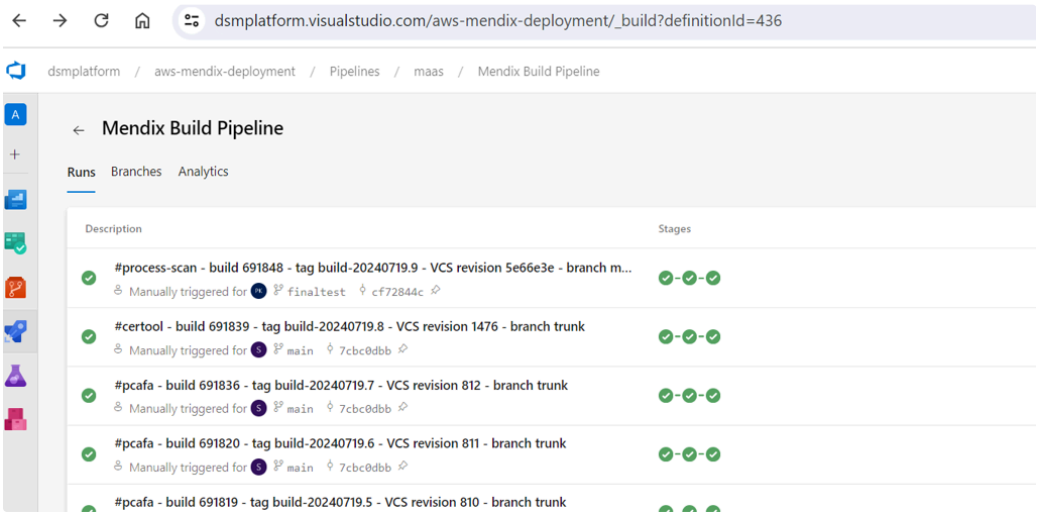


If the pod state is `ImagePullBack`, we need to rebuild and redeploy the application

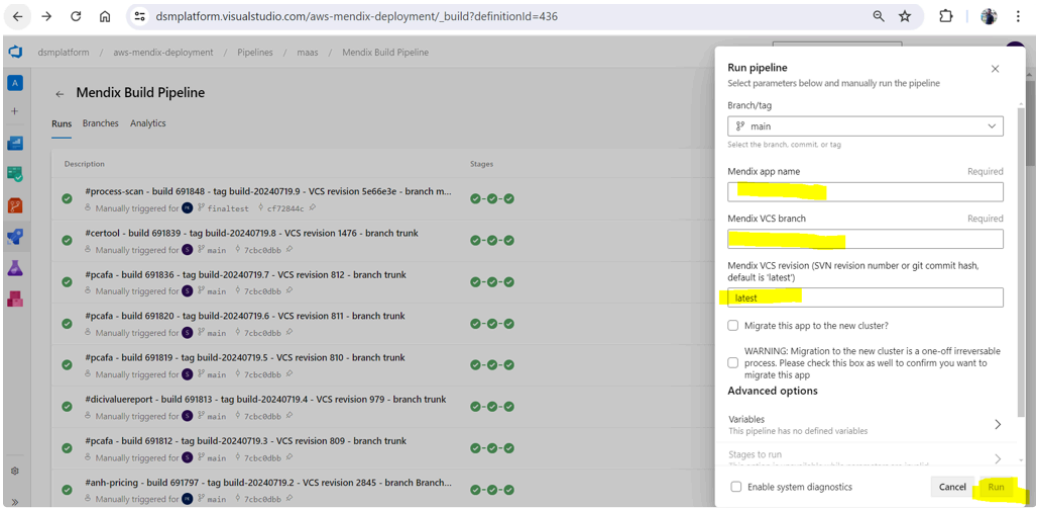
To redeploy the application, log in to the Microsoft Azure DevOps tool



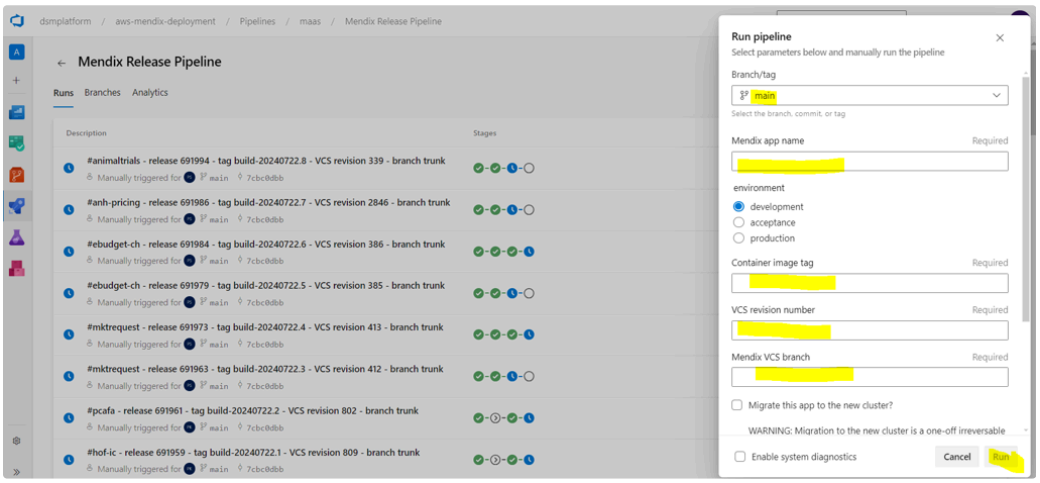
Click on the pipeline and then the Mendix Build Pipeline.



Build the latest commit that was previously deployed in the particular environment



Once the build is successfully completed, the release pipeline will create and deploy the application in the specified environment.



If the application pods are in running status, validate the application URL.

```
CORPAD+834556@DWP-5CD151G4ML MINGW64 ~
$ kubectl get pods -n mendix |grep wrhousestock-vn
wrhousestock-vn-follower-5d5494ff85-z2cw8   2/2     Running   64 (3h4m ago)    5h33m
wrhousestock-vn-leader-dcc59c797-r85kd      2/2     Running   64 (3h3m ago)    5h33m
```

Lastly, ask the application functional team to validate the functionality.