**Lab Steps**

Task 1: Sign in to AWS Management Console

1. Click on the  button, and you will get redirected to AWS Console in a new browser tab.
2. On the AWS sign-in page,
   * Leave the Account ID as default. Never edit/remove the 12 digit Account ID present in the AWS Console. otherwise, you cannot proceed with the lab.
   * Now copy your **User Name** and **Password** in the Lab Console to the **IAM Username and Password** in AWS Console and click on the **Sign in** button.
3. Once Signed In to the AWS Management Console, Make the default AWS Region as **US East (N. Virginia) us-east-1.**

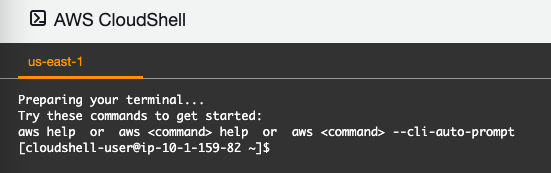
Task 2: Create an Environment in CloudShell

1. Make sure you are in the **N.Virginia** Region.
2. Click on  icon (Cloud Shell) on the top right AWS menu bar.
3. A new tab in your browser opens and if you see a welcome message to cloud shell then click on the **Close** button in that message.
4. You will see a creating environment message on the screen.

Graphical user interface

Description automatically generated with medium confidence

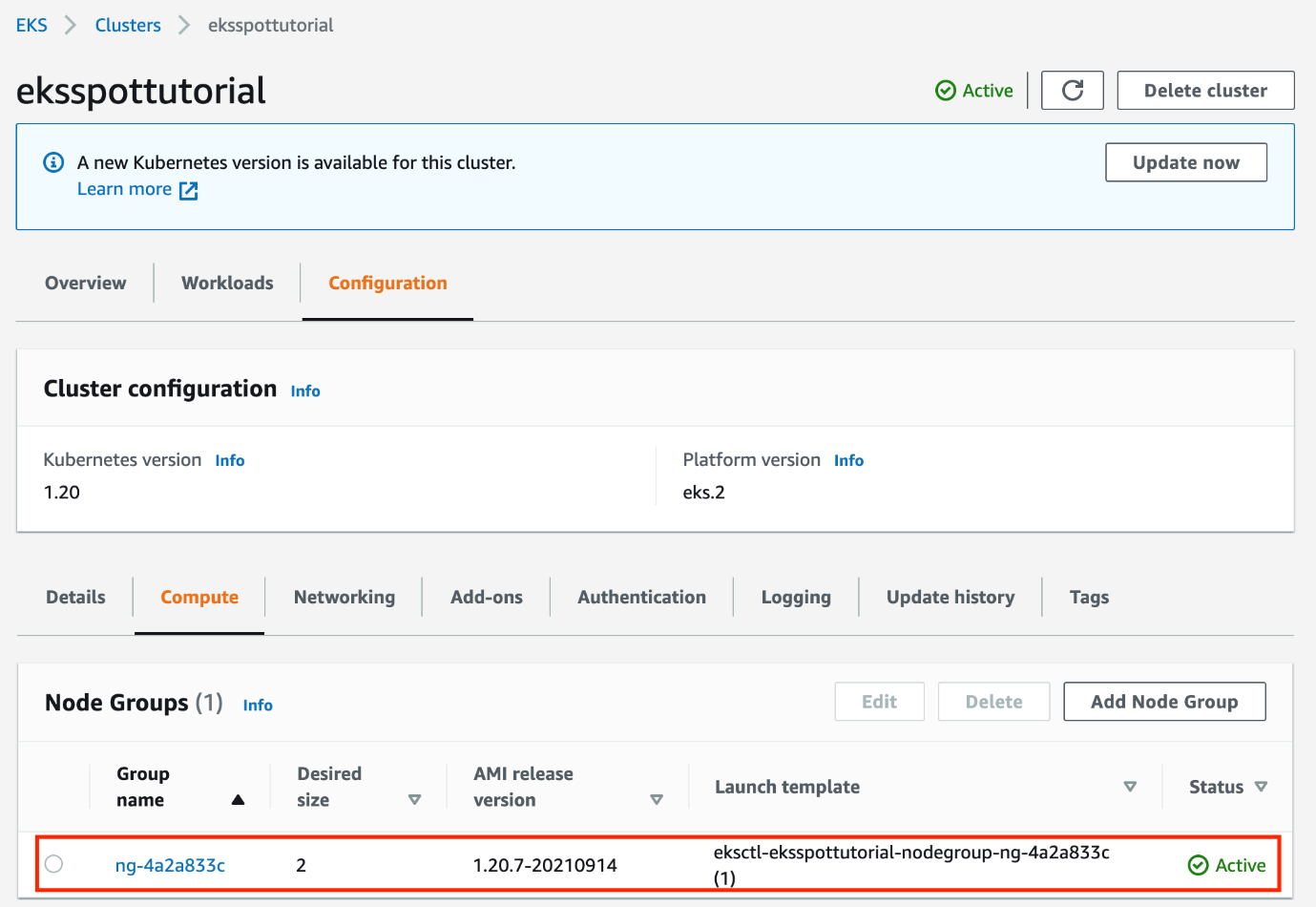
1. Wait for a few minutes to complete the environment creation. Once the environment is created , You are ready to use the terminal.



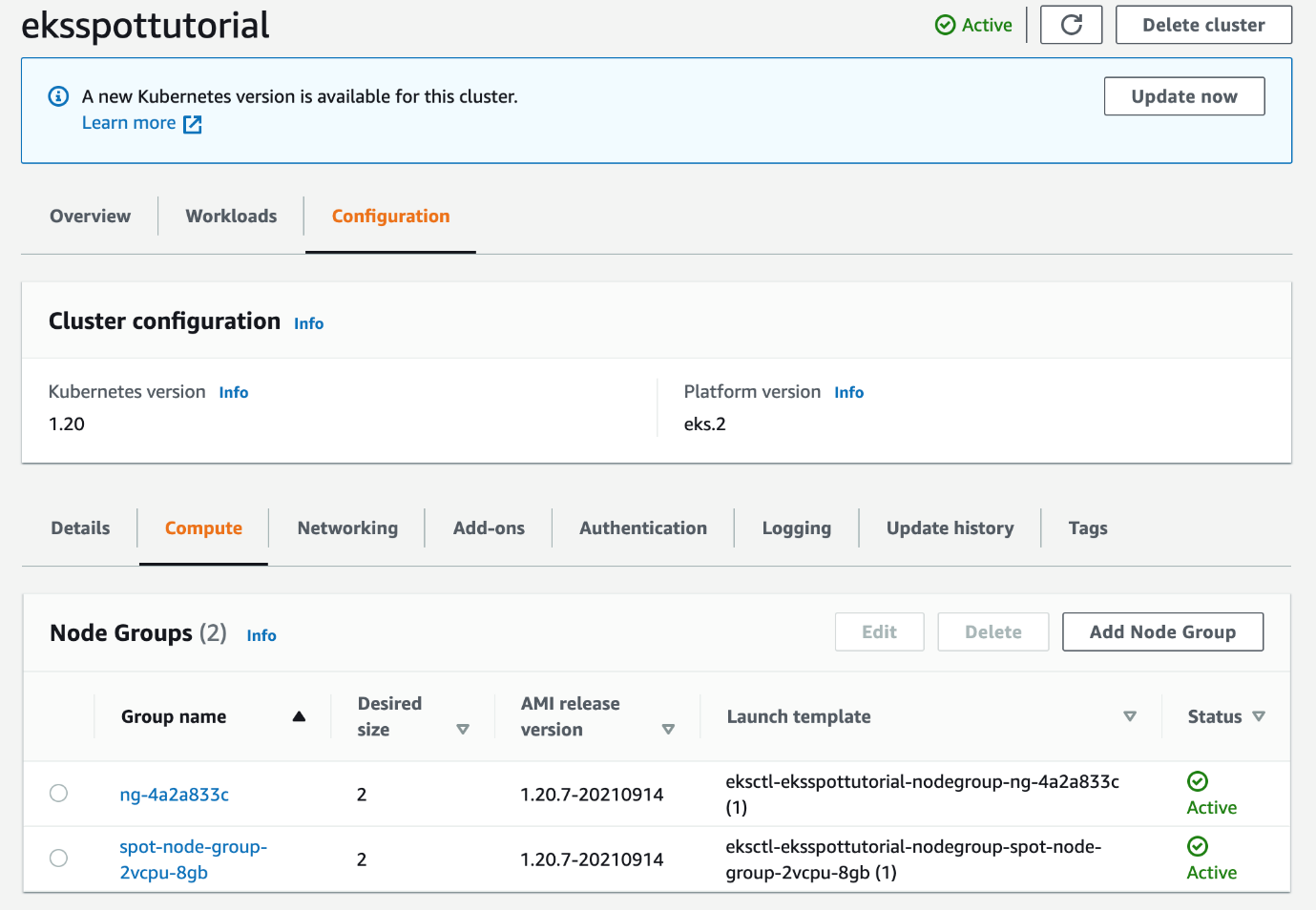
Task 3: Install AWS CLI, eksctl and kubectl

1. Wait until the environment is ready.
2. To install AWS CLI using yum, run the below command on the shell:
   * sudo yum install awscli -y
3. Now check whether the version and confirm the installation:
   * aws --version
4. To install eksctl, copy and paste the below two commands on the shell:
   * curl --silent --location "https://github.com/weaveworks/eksctl/releases/latest/download/eksctl\_$(uname -s)\_amd64.tar.gz" | tar xz -C /tmp
   * sudo mv /tmp/eksctl /usr/local/bin
5. Check the version of eksctl to confirm the installation.
   * eksctl version
6. Once the environment is ready on CloudShell, Download the Amazon EKS vended kubectl binary for your cluster's Kubernetes version from Amazon S3. To do so, run the following command:
   * curl -o kubectl https://amazon-eks.s3.us-east-1.amazonaws.com/1.18.9/2020-11-02/bin/linux/amd64/kubectl
7. Apply execute permissions to the binary.
   * chmod +x ./kubectl
8. Copy the binary to a folder in your PATH. If you have already installed a version of kubectl, then we recommend creating a $HOME/bin/kubectl and ensuring that $HOME/bin comes first in your $PATH.
   * mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$PATH:$HOME/bin
9. After you install kubectl , you can verify its version with the following command:
   * kubectl version --short --client

Task 4: Create an EKS Cluster

1. To create a EKS Cluster, you will use eksctl, run the following command:
   * eksctl create cluster --version=1.20 --name=eksspottutorial --nodes=2 --managed --region=us-east-1 --zones us-east-1a,us-east-1b,us-east-1c --node-type t2.medium --asg-access
2. **Note: Don't close the CloudShell tab.**
3. This command will take around 20-25 minutes to complete the creation of the EKS Cluster.
4. **Note:** There is a possibility that the connection may get lost, you can check the CloudFormation stack.  
   
5. Meanwhile, you can check the EKS Cluster creation status and other events performed by eksctl in the CloudFormation console.
6. Navigate to **CloudFormation**. Click , click on  in the  section.
7. On the left sidebar, click on the **stacks**
8. You will be able to see the new CloudFormation stack getting created and have status as **CREATE\_IN\_PROGRESS.**
9. Click on the stack name **eksctl-eksspottutorial-cluster** and explore the **events** and **resources** tab.
10. Since EKS Cluster will be launched in the new VPC, all the components of it's will be created and configured.
11. Once the VPC resources are created, you will be able to see EKS Cluster creation.
12. Let's check out the current status of the EKS Cluster.
13. Navigate to  by clicking on the  menu available under the  section.
14. On the left sidebar, click on the **clusters,**present under Amazon EKS.
15. You will be able to see an EKS Cluster named **eksspottutorial**.
16. To open the clusters, click on the cluster name and check out the nodes.
17. Switch to the configuration tab and computer sub-tab.  
    
18. Currently, one node group is present with the desired size of 2.
19. To verify the creation of the EKS Cluster, run the following command:
    * kubectl get nodes

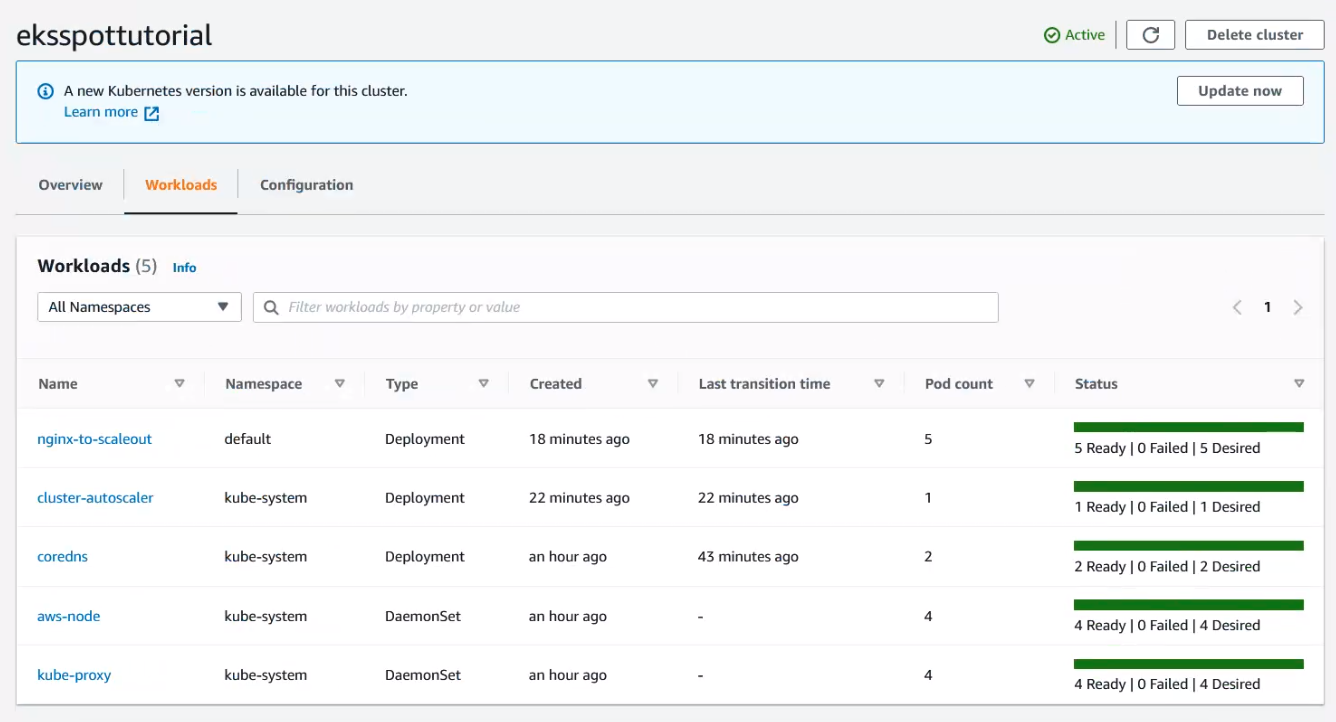
Task 5: Create spot managed node groups

1. To create node groups, run the following command:
   * eksctl create nodegroup --cluster=eksspottutorial --managed --region=us-east-1 --spot --name=spot-node-group-2vcpu-8gb --instance-types=t2.small,t2.micro,t2.medium --nodes-min=2 --nodes-max=5 --asg-access
2. The above command will take around 10 minutes to add the spot-managed node groups to your EKS Cluster.
3. To view the nodes, run the below command:
   * kubectl get nodes --show-labels --selector=eks.amazonaws.com/capacityType=SPOT | grep SPOT
4. Navigate to  by clicking on the  menu available under the  section.
5. On the left sidebar, click on the **clusters.**
6. To open the clusters, click on the cluster name and check out the nodes.
7. Switch to the configuration tab and compute the sub-tab.  
   

Task 6: Deploy the Kubernetes Cluster Autoscaler

1. To download the YAML file, required for deploying the latest version of Kubernetes cluster autoscaler, run the below command:
   * curl -LO https://raw.githubusercontent.com/kubernetes/autoscaler/master/cluster-autoscaler/cloudprovider/aws/examples/cluster-autoscaler-autodiscover.yaml
2. To update the tags for Cluster autoscaler and update the version of Cluster autoscaler, run the following command:
   * sed -i 's/eksspottutorial/eksspottutorial/g' cluster-autoscaler-autodiscover.yaml
   * sed -i 's/v1.17.3/v1.20.0/g' cluster-autoscaler-autodiscover.yaml
3. Deploy the Cluster Autoscaler:
   * kubectl apply -f cluster-autoscaler-autodiscover.yaml

Task 7: Deploy the sample app

1. To create a nginx-to-scaleout.yaml file, run the vi command:
   * touch nginx-to-scaleout.yaml
2. Copy the S3 Bucket starting with name whizlabs, by listing all:
   * aws s3 ls
3. Paste the contents in nginx-to-scaleout.yaml using S3 Bucket, replace the copied bucket name with one below:
   * aws s3 cp s3://<bucket-name>/nginx-to-scaleout.yaml .
4. Deploy the deployment file and confirm the deployment:
   * kubectl apply -f nginx-to-scaleout.yaml
   * kubectl get deployment/nginx-to-scaleout
5. Scale the deployment, by running 5 replicas of it:
   * kubectl scale --replicas=5 deployment/nginx-to-scaleout
6. Check the deployed pods, some of them may have a status as **Pending**:
   * kubectl get pods
7. Navigate to  by clicking on the  menu available under the  section.
8. On the left sidebar, click on the **clusters.**
9. To open the clusters, click on the cluster name and check out the nodes.
10. Switch to the **Workloads** tab.  
    
11. Confirm that all the pods are scheduled:
    * kubectl get pods

Task 8: Delete AWS Resources

1. Delete the NGINX web server deployed:
   * kubectl delete -f nginx-to-scaleout.yaml
2. Remove the EKS Spot managed node groups:
   * eksctl delete nodegroup --cluster=eksspottutorial --name=spot-node-group-2vcpu-8gb
3. Delete the EKS Cluster:
   * eksctl delete cluster --name eksspottutorial
   * **Note:**It may take up to 10 minutes to delete all the resources.

Deleting the CloudShell directory

1. Navigate to the CloudShell tab.
2. Click on the  button on top right corner and select .
3. Enter **delete** in the text input field and click on the  button and close the browser tab.

**Completion and Conclusion**

1. You have successfully created and launched Amazon EKS Cluster.
2. You have successfully installed AWS CLI, Kubectl and eksctl in AWS Cloudshell.
3. You have successfully configured AWS Cloudshell to communicate with AWS EKS Cluster.

**End Lab**