Use RKE to create a k8s cluster and connect to Jenkins

Step 1: Create RKE User and Group on All Nodes

1. **Create the RKE user and group** on all nodes in the Kubernetes cluster:

   ```
   sudo groupadd rke sudo useradd -g rke rke
   ```

2. **Add the user to the Docker group**:

   ```
   sudo usermod -aG docker rke
   ```

3. **Repeat the steps** on your local machine as well.

Step 2: SSH Key Exchange

1. **Generate SSH keys**:

   ```
   ssh-keygen -t rsa
   ```

2. **Copy the SSH public key** to all nodes for passwordless SSH access:

   ```
   ssh-copy-id -i ~/.ssh/id_rsa.pub rke@rke_host ssh-copy-id -i ~/.ssh/id_rsa.pub rke@host(1-n)
   ```

Step 3: Install RKE (Rancher Kubernetes Engine)

1. **Download the latest RKE binary** from the official RKE GitHub Releases.
   ```
   curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
   ```
2. **Make the binary executable**:
   ```
   chmod +x kubectl
   ```
   ```
   sudo mv kubectl /usr/local/bin/
   ```

Step 4: Set Proxy Configuration

1. **Set the `NO_PROXY` environment variable** to include your RKE host and other nodes:

   ```
   export NO_PROXY=rke_host,host(1-n)
   ```

Step 5: Configure Kubernetes Cluster

1. **Create a YAML configuration file** (`cluster.yml`). You can find sample configuration files at Rancher Documentation.
2. **Run the `rke up` command** to initiate the Kubernetes cluster setup:

   ```
   rke up --config cluster.yml
   ```

   After a successful setup, you should see:

   ```
   INFO[0272] Finished building Kubernetes cluster successfully
   ```

Step 6: Access the Kube Config File

1. **Copy the generated kube_config_cluster.yml** to your local machine:

   ```
   mkdir -p ~/.kube && cp kube_config_cluster.yml ~/.kube/config
   ```

2. **Set the `KUBECONFIG` environment variable** to point to your kube config file:

   ```
   export KUBECONFIG=/home/rke/.kube/config
   ```

3. **Add the `NO_PROXY` environment variable** to your `.bashrc` file to ensure proper proxy configuration (should already be there from step 4.1):

   ```
   export NO_PROXY=rke_host,host(1-n)
   ```

4. **To access the cluster with K9s**, you can set up an alias:

   ```
   alias k9s='docker run --rm -it -v $KUBECONFIG:/root/.kube/config quay.io/derailed/k9s'
   ```

Step 7: Verify Cluster Nodes

1. **Verify the Kubernetes nodes**:

   ```
   kubectl get nodes
   ```

   This should show the nodes you added in the `cluster.yml` file.

Step 8: Add Kubernetes Credentials to Jenkins

1. **Add credentials to Jenkins** using the `kube_config_cluster.yml` file:
   - Go to **Manage Jenkins → Configure System**.
   - Add **credentials of type "Secret File"**, and upload the `~/.kube/config` file.
2. **Configure Jenkins Cloud Settings**:
   - Go to **Manage Jenkins → Configure Clouds**.

- Add a new cloud configuration and base it on your existing cloud configuration, changing the Kubernetes cloud name to match the one you created (e.g., `kubernetes_test`). At the end of this doc is a screen shot of the configuration.

Step 9: Run Pipeline to Test the Kubernetes Cloud

1. **Test the new cloud** by running a pipeline.

- Ensure that you change the cloud name and the labels:
  - Label in cluster as in pipeline
    - `kubectl label --list nodes <node_name>`
    - `kubectl label nodes <node_name> workload=<test_cluster>`
    - `pipeline { agent { kubernetes { .... usefor:workload ... values: - test_cluster`
  - Ensure that you change the *cloud name* in the pipeline script:

    `pipeline { agent { kubernetes { cloud '<kubernetes_test>'`

2. **Verify pod creation**:
   - Before running the pipeline, use:

     `kubectl get pods`

     This should show no pods.
   - After running the pipeline, verify that a new pod has been created:

     `kubectl get pods`
   - Troubleshoot the pod:
     `kubectl describe pod_name`

Step 10: Install crowdstrike agent through a daemon set

make sure there is not a flacon service on the cluster's nodes.

`systemctl | grep falcon`
`systemctl stop falcon-sensor`
`systemctl status falcon-sensor`

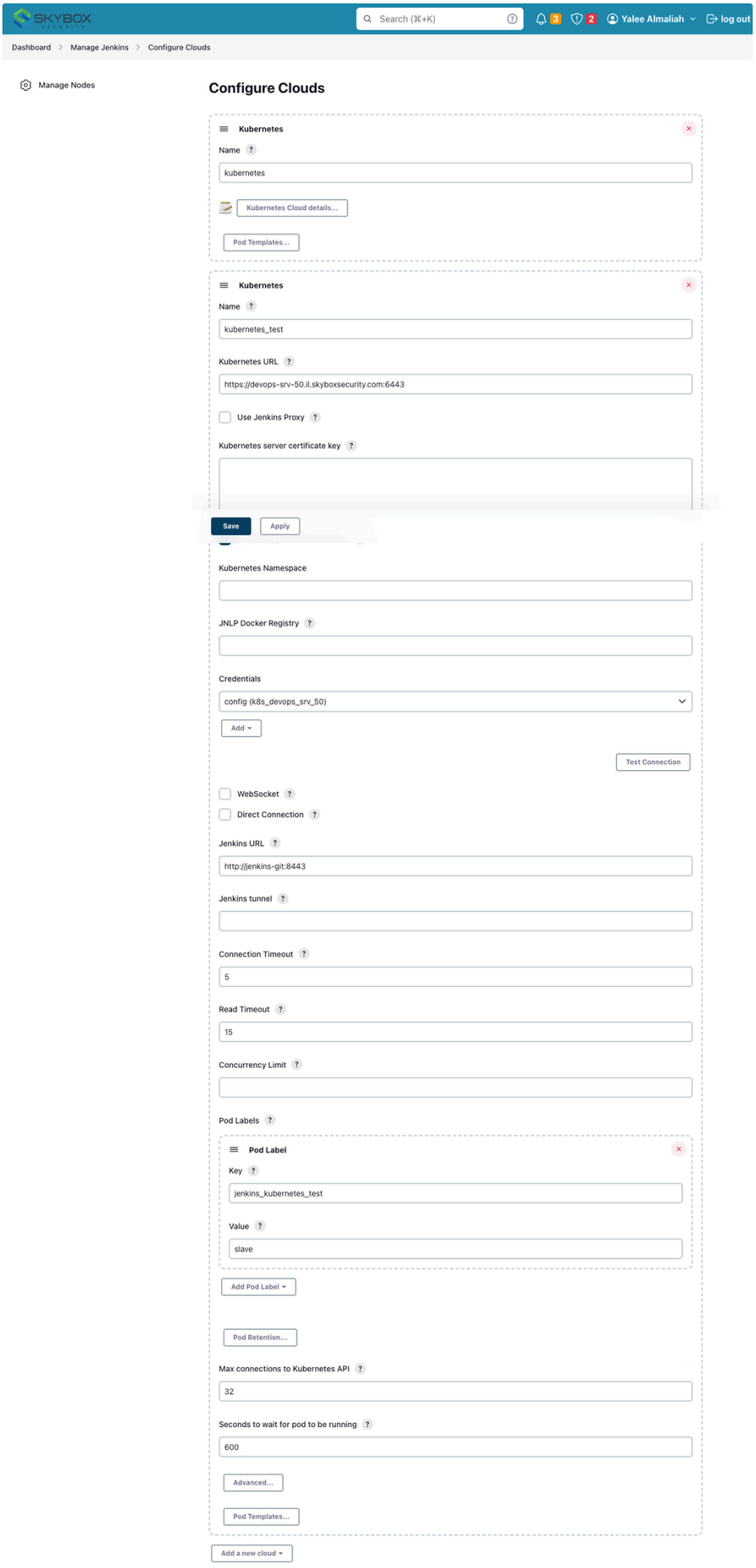
this script is based on mt-deployment/helm/falcon/install.sh

```
# Define environment variables
export FALCON_IMAGE_TAG="falcon-sensor-7.18.0-US-1"
export FALCON_IMAGE_REPO="docker-nexus3.il.skyboxsecurity.com/devops/agent"
export FALCON_SENSOR_REPO="crowdstrike/falcon-sensor"
export FALCON_NAMESPACE="falcon-system"
export FALCON_CID="___" # Retrieved from consul aws/jenkins_global_config/on_prem_ec2_crowdStrike_client_cid
export FALCON_TAG="__" # Retrieved from consul aws/jenkins_global_config/on_prem_ec2_crowdStrike_falcon_tag

# Add CrowdStrike Helm repository
helm repo add crowdstrike https://crowdstrike.github.io/falcon-helm
helm repo update

# Create and label the namespace
kubectl create ns ${FALCON_NAMESPACE}
kubectl label ns --overwrite ${FALCON_NAMESPACE} pod-security.kubernetes.io/enforce=privileged
kubectl label ns --overwrite ${FALCON_NAMESPACE} pod-security.kubernetes.io/audit=privileged
kubectl label ns --overwrite ${FALCON_NAMESPACE} pod-security.kubernetes.io/warn=privileged

# Deploy or upgrade the Falcon sensor
helm upgrade --install falcon-sensor ${FALCON_SENSOR_REPO} -n ${FALCON_NAMESPACE} \
  --set falcon.cid=${FALCON_CID} \
  --set node.image.repository=${FALCON_IMAGE_REPO} \
  --set node.image.tag=${FALCON_IMAGE_TAG} \
  --set falcon.tags=${FALCON_TAG}
```

Step 11: Cleanup (Optional)

1. **To remove a Kubernetes cluster** created with RKE, run:

   `rke remove --config cluster.yml`


SKYBOX SECURITY
Search (⌘+K)
3
2
Yalee Almaliah
log out
Dashboard
Manage Jenkins
Configure Clouds
Manage Nodes
Configure Clouds
Kubernetes
Name
kubernetes
Kubernetes Cloud details...
Pod Templates...
Kubernetes
Name
kubernetes_test
Kubernetes URL
https://devops-srv-50.il.skyboxsecurity.com:6443
Use Jenkins Proxy
Kubernetes server certificate key
Save
Apply
Kubernetes Namespace
JNLP Docker Registry
Credentials
config (k8s_devops_srv_50)
Add
Test Connection
WebSocket
Direct Connection
Jenkins URL
http://jenkins-git:8443
Jenkins tunnel
Connection Timeout
5
Read Timeout
15
Concurrency Limit
Pod Labels
Pod Label
Key
jenkins_kubernetes_test
Value
slave
Add Pod Label
Pod Retention...
Max connections to Kubernetes API
32
Seconds to wait for pod to be running
600
Advanced...
Pod Templates...
Add a new cloud