Kubernetes Automation Script Documentation

This script automates common Kubernetes setup tasks, including installing Helm and KEDA, setting up the Metrics Server, deploying applications, and configuring KEDA autoscaling. Designed to simplify deployment management, it supports both automated and interactive workflows.

Table of Contents

Prerequisites

Ensure the following are installed and configured before running the script:

- Kubernetes CLI (kubectl): Required to interact with your Kubernetes cluster.
- Bash: Script is written in Bash and should be run in a Unix-based environment.
- Internet Connection: Required to download Helm and other resources.
- Sudo Access: Some commands may require elevated permissions.

Installation

1. Download the script file:
2. Make the script executable:

```
chmod +x kubernetes-automation-script.sh
```

3. Run the script using the help menu to view available options:

```
./kubernetes-automation-script.sh --help
```

Script Usage

The script can be run in two modes:

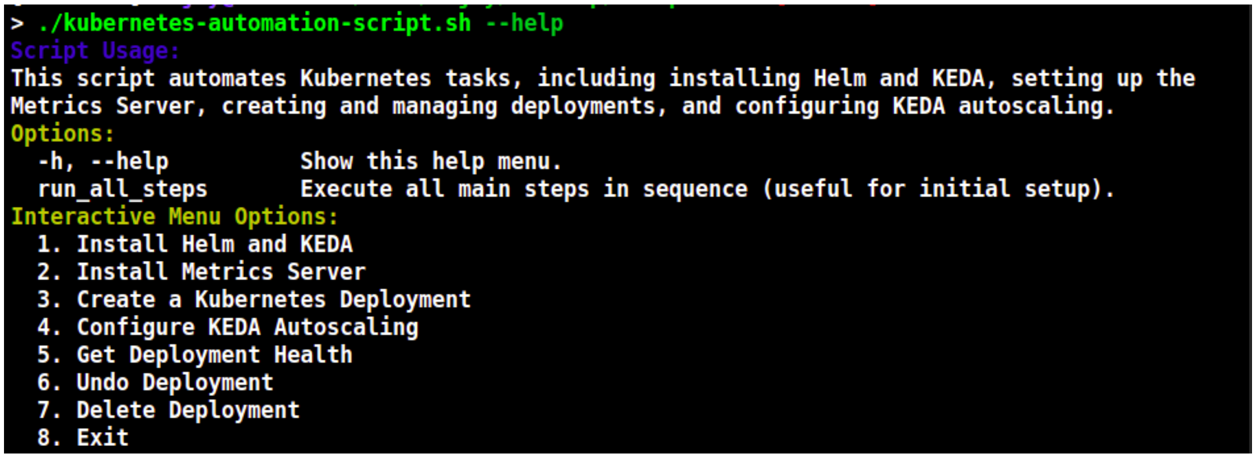
1. **Automated Run**: Run all major steps in sequence.
2. **Interactive Mode**: Execute specific functions through a menu.

Help Menu

To access the help menu, run:

```
./kubernetes-automation-script.sh --help
```

This displays details on each option, including the steps covered in the automated run and the interactive mode functions.



```
> ./kubernetes-automation-script.sh --help
Script Usage:
This script automates Kubernetes tasks, including installing Helm and KEDA, setting up the
Metrics Server, creating and managing deployments, and configuring KEDA autoscaling.
Options:
  -h, --help          Show this help menu.
  run_all_steps       Execute all main steps in sequence (useful for initial setup).
Interactive Menu Options:
  1. Install Helm and KEDA
  2. Install Metrics Server
  3. Create a Kubernetes Deployment
  4. Configure KEDA Autoscaling
  5. Get Deployment Health
  6. Undo Deployment
  7. Delete Deployment
  8. Exit
```

Running All Steps

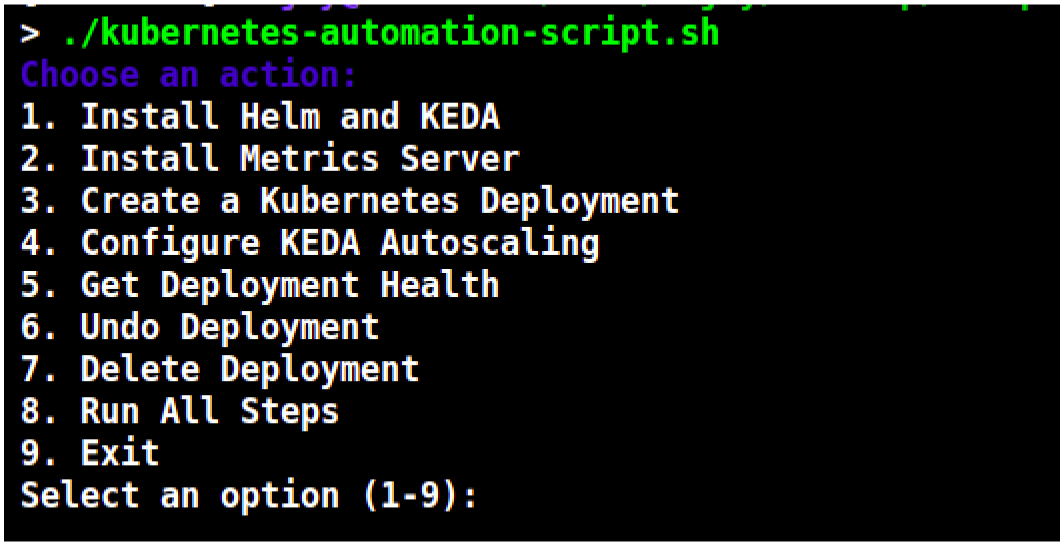
To execute all main tasks (installing Helm, KEDA, Metrics Server, creating deployments, and configuring autoscaling) in one run, use:

```
./kubernetes-automation-script.sh run_all_steps
```

Interactive Menu

To run tasks selectively, start the interactive menu by running the script without arguments:

```
./kubernetes-automation-script.sh
```



```
> ./kubernetes-automation-script.sh
Choose an action:
1. Install Helm and KEDA
2. Install Metrics Server
3. Create a Kubernetes Deployment
4. Configure KEDA Autoscaling
5. Get Deployment Health
6. Undo Deployment
7. Delete Deployment
8. Run All Steps
9. Exit
Select an option (1-9):
```

Function Descriptions

Install Helm and KEDA

- **Purpose**: Install Helm (if not already installed) and deploy KEDA in the `keda` namespace.
- **Command**: Runs in the background if `run_all_steps` is chosen or selected from the interactive menu.

Install Metrics Server

- **Purpose**: Install Metrics Server with custom arguments to allow insecure connections to kubelets.
- **Details**: Downloads the latest `components.yaml` for the Metrics Server, adds `--kubelet-insecure-tls` flag, and applies the configuration to the cluster.

Create Kubernetes Deployment

- **Purpose**: Creates a Kubernetes deployment and service (NodePort) with user-specified options.
- **Input Options**:
  - **Namespace**: Namespace for the deployment.
  - **Deployment Name**: Name of the deployment.
  - **Container Image**: Image for the container (default: `nginx`).
  - **Port and Resource Limits**: Configure CPU and memory limits/requests.

Configure KEDA Autoscaling

- **Purpose**: Set up autoscaling for a deployment using KEDA, based on CPU utilization or custom triggers.
- **Input**:
  - **Deployment Name**: Name of the deployment to scale.
  - **Namespace**: Deployment namespace.
  - **Target Utilization**: CPU target percentage for scaling.

Get Deployment Health

- **Purpose**: Retrieve and display the health status of a specified deployment, including resource usage if Metrics Server is enabled.

Undo Deployment

- **Purpose**: Roll back the most recent deployment for a specified resource.
- **Input**:
  - **Deployment Name**: Name of the deployment to undo.
  - **Namespace**: Namespace of the deployment.

Delete Deployment

- **Purpose**: Permanently delete a specified deployment and its resources.
- **Input**:
  - **Deployment Name**: Name of the deployment to delete.
  - **Namespace**: Namespace of the deployment.

Examples

1. **Run All Steps in Sequence**:

```
./kubernetes-automation-script.sh run_all_steps
```

2. **Create a Deployment with Default Options**: From the interactive menu, choose `Create Kubernetes Deployment`, and enter deployment specifics when prompted.
3. **Configure KEDA Autoscaling**: From the interactive menu, choose `Configure KEDA Autoscaling`, and specify the deployment, namespace, and target CPU utilization.

Troubleshooting

Helm or KEDA Not Installing

- Ensure your internet connection is stable, and verify cluster connectivity. Check Helm documentation for alternative installation options.

Metrics Server Fails to Start

- Make sure the `components.yaml` is downloaded correctly. The `--kubelet-insecure-tls` argument is required for many Kubernetes setups with self-signed certificates.

Deployment Issues

If a deployment fails:

- Verify the namespace and deployment names.
- Ensure the container image is available in your registry.

No Resource Usage Data

- If `kubectl top` commands are unresponsive, check that the Metrics Server is running. The Metrics Server may take a few moments to initialize after installation.

#k8s_script.sh

```bash
#!/bin/bash

# Color codes
RED='\033[0;31m'
GREEN='\033[0;32m'
YELLOW='\033[0;33m'
BLUE='\033[0;34m'
NC='\033[0m' # No Color

# Print a breakline for readability
print_breakline() {
      echo -e "${YELLOW}============================================================${NC}"
}

# Set default Kubernetes context
## $ kubectl config get-contexts
KUBECTL_CONTEXT="kind-devops-test"

# Initial prerequisites check
check_prerequisites() {
      echo -e "${BLUE}Checking prerequisites...${NC}"
      check_k8s_connection
}

# Check if connected to the Kubernetes cluster
check_k8s_connection() {
      current_context=$(kubectl config current-context 2>/dev/null)

      if [[ "$current_context" != "$KUBECTL_CONTEXT" ]]; then
      echo -e "${YELLOW}>>> Not connected to the Kubernetes cluster. Please provide the path to your kubeconfig file to connect.${NC}"

      # Prompt user for kubeconfig path
      read -p "Enter the path to your kubeconfig file: " kubeconfig_path

      # Attempt to use the provided kubeconfig
      export KUBECONFIG="$kubeconfig_path"
      kubectl config use-context "$KUBECTL_CONTEXT" &> /dev/null

      # Check again if the connection was successful
      if [[ "$(kubectl config current-context)" == "$KUBECTL_CONTEXT" ]]; then
```

```
            echo -e "${GREEN}Connected to Kubernetes cluster with context:
$KUBECTL_CONTEXT${NC}"
      else
            echo -e "${RED}Failed to connect to Kubernetes cluster with the provided
kubeconfig file. Please verify the file and try again.${NC}"
            exit 1
      fi
      else
      echo -e "${GREEN}Already connected to Kubernetes cluster with context:
$KUBECTL_CONTEXT${NC}"
      fi
}

# Help menu function
print_help() {
      echo -e "${BLUE}Script Usage:${NC}"
      echo "This script automates Kubernetes tasks, including installing Helm and KEDA,
setting up the Metrics Server, creating and managing deployments, and configuring KEDA
autoscaling."
      echo -e "${YELLOW}Options:${NC}"
      echo "  -h, --help            Show this help menu."
      echo "  run_all_steps         Execute all main steps in sequence (useful for initial
setup)."
      echo -e "${YELLOW}Interactive Menu Options:${NC}"
      echo "  1. Install Helm and KEDA"
      echo "  2. Install Metrics Server"
      echo "  3. Create a Kubernetes Deployment"
      echo "  4. Configure KEDA Autoscaling"
      echo "  5. Get Deployment Health"
      echo "  6. Undo Deployment"
      echo "  7. Delete Deployment"
      echo "  8. Exit"
}

# Run all main steps sequentially
run_all_steps() {
      echo -e "${BLUE}Running all setup steps in sequence...${NC}"
      install_helm_and_keda
      install_metrics_server
      create_deployment
      configure_keda_autoscaling
      echo -e "${GREEN}All steps completed successfully.${NC}"
}
```

```
# Function to install Helm and KEDA
install_helm_and_keda() {
      if ! command -v helm &> /dev/null; then
      print_breakline
      echo -e "${YELLOW}>>> Installing Helm...${NC}"
      curl -fsSL https://raw.githubusercontent.com/helm/helm/master/scripts/get-helm-3 |
bash
      echo -e "${GREEN}Helm installed successfully.${NC}"
      else
      echo -e "${GREEN}Helm is already installed.${NC}"
      fi

      if ! kubectl get ns keda &> /dev/null; then
      print_breakline
      echo -e "${YELLOW}>>> Installing KEDA...${NC}"
      helm repo add kedacore https://kedacore.github.io/charts
      helm repo update
      helm install keda kedacore/keda --namespace keda --create-namespace
      echo -e "${GREEN}KEDA installed successfully.${NC}"
      else
      echo -e "${GREEN}KEDA is already installed.${NC}"
      fi
}

# Function to install the Metrics Server with custom arguments
install_metrics_server() {
      if ! kubectl get deployment metrics-server -n kube-system &> /dev/null; then
      print_breakline
      echo -e "${YELLOW}>>> Installing Metrics Server...${NC}"
      wget
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml
      sed -i '/args:/a\       - --kubelet-insecure-tls' components.yaml
      kubectl apply -f components.yaml
      rm components.yaml
      echo -e "${GREEN}Metrics Server installed successfully with custom arguments.${NC}"
      else
      echo -e "${GREEN}Metrics Server is already installed.${NC}"
      fi
}

# Function to create a Kubernetes deployment
create_deployment() {
      print_breakline
      echo -e "${BLUE}>>> Creating a new deployment...${NC}"

      read -p "Namespace (default is 'default'): " namespace
      namespace=${namespace:-default}

      read -p "Deployment name (default is 'my-deployment'): " deployment_name
      deployment_name=${deployment_name:-my-deployment}

      read -p "Container image (default is 'nginx'): " image
      image=${image:-nginx}

      read -p "Container port (default 80): " container_port
      container_port=${container_port:-80}

      read -p "CPU request (default is '100m'): " cpu_request
      cpu_request=${cpu_request:-100m}

      read -p "Memory request (default is '128Mi'): " memory_request
      memory_request=${memory_request:-128Mi}

      read -p "CPU limit (default is '200m'): " cpu_limit
      cpu_limit=${cpu_limit:-200m}

      read -p "Memory limit (default is '256Mi'): " memory_limit
      memory_limit=${memory_limit:-256Mi}

      print_breakline
      echo -e "${YELLOW}Creating namespace ${namespace}...${NC}"
      kubectl create namespace "$namespace" --dry-run=client -o yaml | kubectl apply -f -

      print_breakline
      echo -e "${YELLOW}Creating deployment $deployment_name...${NC}"
      kubectl create deployment "$deployment_name" --image="$image" -n "$namespace"

      kubectl set resources deployment "$deployment_name" -n "$namespace" \
      --requests=cpu="$cpu_request",memory="$memory_request" \
      --limits=cpu="$cpu_limit",memory="$memory_limit"

      print_breakline
      echo -e "${YELLOW}Creating NodePort service for $deployment_name...${NC}"
      kubectl expose deployment "$deployment_name" --type=NodePort
--name="${deployment_name}-service" -n "$namespace" --port=80
```

```
--target-port="$container_port"

      echo -e "${GREEN}Deployment $deployment_name created successfully.${NC}"
      print_breakline
}

# Function to configure KEDA autoscaling
configure_keda_autoscaling() {
      print_breakline
      echo -e "${BLUE}>>> Configuring KEDA Autoscaling...${NC}"

      read -p "Enter the deployment name for autoscaling: " deployment_name
      read -p "Enter the namespace of the deployment: " namespace
      read -p "Enter the target memory utilization percentage (e.g., 50): " target_value
      target_value="${target_value}%"

      cat <<EOF > keda_scaled_object.yaml
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: ${deployment_name}-scaledobject
  namespace: ${namespace}
spec:
  maxReplicaCount: 5
  minReplicaCount: 1
  scaleTargetRef:
      name: ${deployment_name}
  triggers:
      - type: cpu
      metricType: Utilization
      metadata:
      value: "${target_value}"
EOF

      kubectl apply -f keda_scaled_object.yaml -n "$namespace"
      echo -e "${GREEN}KEDA Autoscaling configured successfully for deployment
${deployment_name}.${NC}"
      print_breakline
}

# Function to get the health status of a deployment
get_deployment_health() {
      print_breakline
      echo -e "${BLUE}>>> Get Deployment Health...${NC}"
      read -p "Enter the namespace of the deployment: " namespace
      read -p "Enter the deployment name: " deployment_name

      echo -e "${YELLOW}Fetching health status for deployment ${deployment_name}...${NC}"
      kubectl get deployment "$deployment_name" -n "$namespace" -o wide

      if ! kubectl top nodes &> /dev/null; then
      echo -e "${RED}Metrics Server is not installed. Skipping resource usage display.${NC}"
      else
      echo -e "${YELLOW}Resource usage:${NC}"
      kubectl top pods -n "$namespace" | grep "$deployment_name" || echo -e "${RED}No
resource usage data available.${NC}"
      fi
      print_breakline
}

# Function to undo a deployment
undo_deployment() {
      print_breakline
      echo -e "${BLUE}>>> Undo Deployment Process...${NC}"
      read -p "Enter the deployment name to undo: " deployment_name
      read -p "Enter the namespace of the deployment: " namespace

      echo -e "${YELLOW}Undoing the last deployment for ${deployment_name}...${NC}"
      kubectl rollout undo deployment "$deployment_name" -n "$namespace"
      echo -e "${GREEN}Deployment ${deployment_name} undone successfully.${NC}"
      print_breakline
}

# Function to delete a deployment
delete_deployment() {
      print_breakline
      echo -e "${BLUE}>>> Deleting Deployment...${NC}"
      read -p "Enter the deployment name to delete: " deployment_name
      read -p "Enter the namespace of the deployment: " namespace

      echo -e "${YELLOW}Deleting deployment ${deployment_name}...${NC}"
      kubectl delete deployment "$deployment_name" -n "$namespace"
      echo -e "${GREEN}Deployment ${deployment_name} deleted successfully.${NC}"
      print_breakline
}
```

```
# Run prerequisite check before any main actions (Checking K8s connection)
check_prerequisites

# Main script execution
if [[ "$1" == "-h" || "$1" == "--help" ]]; then
      print_help
      exit 0
elif [[ "$1" == "run_all_steps" ]]; then
      run_all_steps
      exit 0
fi

# Interactive menu
while true; do
      echo -e "${BLUE}Choose an action:${NC}"
      echo "1. Install Helm and KEDA"
      echo "2. Install Metrics Server"
      echo "3. Create a Kubernetes Deployment"
      echo "4. Configure KEDA Autoscaling"
      echo "5. Get Deployment Health"
      echo "6. Undo Deployment"
      echo "7. Delete Deployment"
      echo "8. Run All Steps"
      echo "9. Exit"

      read -p "Select an option (1-9): " choice

      case "$choice" in
      1) install_helm_and_keda ;;
      2) install_metrics_server ;;
      3) create_deployment ;;
      4) configure_keda_autoscaling ;;
      5) get_deployment_health ;;
      6) undo_deployment ;;
      7) delete_deployment ;;
      8) run_all_steps ;;
      9) echo -e "${GREEN}Exiting.${NC}"; break ;;
      *) echo -e "${RED}Invalid choice. Please select a valid option.${NC}" ;;
      esac
done
```