SimpliSmart Project Documentation

Project Overview

This project provides a CLI-based automation suite to connect to a Kubernetes cluster, install KEDA using Helm, deploy workloads, and perform health checks. The implementation also includes a Jenkins pipeline for CI/CD integration.

Prerequisites

1. Install Python 3.x on your machine.
   - For Linux/macOS: `sudo apt install python3` or `brew install python`
   - For Windows: Download from Python.org
2. Install `pip` (Python package manager).
   - Verify with `pip --version`, or install using `sudo apt install python3-pip`.
3. Ensure you have working access to a Kubernetes cluster with `kubectl` configured.
4. Ensure Jenkins is installed and configured if running the CI/CD pipeline.

Installation

1. Clone the repository:

```
git clone https://github.com/yeswanth1218/assesment.git
cd assesment
```

2. Install Python dependencies:

```
pip install -r scripts/requirements.txt
```

Project Layout

The repository contains the following files and directories:

1. Repository Structure

```
├── CI-CD/
│   └── CD.groovy
├── scripts/
│   ├── setup_cluster.py
│   ├── installations.py
│   ├── common.py
│   ├── deploy_workload.py
│   ├── health_check.py
│   └── requirements.txt
├── templates/
│   ├── deployment_template.yaml
│   ├── service_template.yaml
│   └── keda_scaledobject_template.yaml
└── .env
```

2. File Descriptions

1. `CI-CD/Jenkinsfile`:
   - Automates the setup, deployment, and health checks through Jenkins stages.
   - Parameters allow users to choose specific actions (e.g., setup, deploy, check health).
2. `scripts/setup_cluster.py`:
   - Sets up the Kubernetes cluster by:
     - Ensuring `kubectl` and `helm` are installed.
     - Validating cluster connectivity.
     - Installing KEDA via Helm.
3. `scripts/deploy_workload.py`:
   - Deploys an application (e.g., NGINX) using:
     - A Deployment YAML for pod configuration.
     - A Service YAML for exposing the application.
     - A KEDA ScaledObject YAML for auto-scaling policies.
4. `scripts/health_check.py`:
   - Verifies the deployment status, pod health, and resource usage (CPU/Memory).
5. `scripts/installations.py`:
   - Contains helper functions to check and install tools like `kubectl` and `helm`.
6. `scripts/common.py`:
   - Provides reusable utilities for running shell commands, validating cluster connectivity, and managing Kubernetes manifests.
7. `templates/`:
   - Contains YAML templates for deployment, service, and scaling configurations, parameterized for reusability.
8. `.env`:
   - Stores environment variables to configure namespace, deployment name, resource limits, and scaling policies.

Usage Journey

This section walks through the usage of the CLI tool with examples and screenshots.

1. Setting Up the Cluster

Run the `setup_cluster.py` script to ensure :

- Checks whether you are connected to a k8s cluster on not
- Checks if `kubectl` and `helm` are installed locally. If not, it will check for the machine OS and version and then install both the tools automatically.
- Installs `keda` using a helm chart

  ```
  python3 scripts/setup_cluster.py
  ```

Reference output:

```
Checking if 'kubectl' and 'helm' are installed...
kubectl is already installed.
helm is already installed.
Current cluster context: my-cluster-context
Adding KEDA Helm chart repository...
KEDA installed successfully in namespace 'keda'.
```

2. Deploying the Workload

Deploy the workload by running `deploy_workload.py`:

```
python3 scripts/deploy_workload.py
```

Screenshot:

3. Performing a Health Check

Run `health_check.py` to verify the workload's status:

```
python3 scripts/health_check.py
```

Reference output:

```
Namespace 'myapp' created.
Deployment 'my-nginx' created in namespace 'myapp'.
Service created and exposed via LoadBalancer.
KEDA ScaledObject configured for auto-scaling.
```

4. CI/CD Automation

To use Jenkins for automating the workflow:

1. Add the `Jenkinsfile` to your Jenkins pipeline configuration.
2. Set the repository URL and configure environment variables.
3. Trigger the pipeline and select the desired action (`Setup Cluster`, `Deploy Workload`, `Health Check`, or `All`).

Additional Notes

1. **Error Handling**:
   - The scripts provide clear error messages for common issues like missing tools or misconfigured cluster contexts.
2. **Scaling Policies/ variable params**:
   - Adjust the `.env` file to modify parameters (e.g., `MIN_REPLICAS`, `MAX_REPLICAS`).