MigrationStrategy.py:

- El script de Python que contiene la estrategia de activación y la predicción de la carga de trabajo.
- USANDO psutil obtenemos los datos de la CPU y predecimos la carga de trabajo y ejecutamos la estrategia de activación.

cpu_util.py:

- Utiliza la biblioteca psutil para obtener los datos de la serie temporal de la CPU para un sistema
- biblioteca psutil invocada

patternMatching.py:

- Paso de preprocesamiento y coincidencia para hacer coincidir los próximos datos de la CPU con los datos existentes
- datos implementados de coincidencia de patrones

trigger_strategy.py:

- Según la carga de trabajo, podemos migrar los contenedores o no
- **trigger_strategy** implementada

dockersetup.sh:

- Pasos de instalación para Docker
- script de configuración de Docker implementado, puede instalar Docker con esto

migrate.sh:

- Según la estrategia de migración, transfiera el contenedor al otro sistema
- edite la dirección IP de la máquina de destino para ejecutarla en otros sistemas.

final.sh:

- Ejecute MigrationStratergy y, según el valor que devuelva, ejecute migrate
- el primer script para ejecutar las otras tareas

run.sh:

- Cargue el contenedor comprimido en la ventana acoplable en la máquina de destino
- el script para cargar el contenedor de la cremallera en la ventana acoplable

EXECUTION STEPS

Paso 1 Ejecute dockersetup.sh en las máquinas de origen y destino. Después de la instalación, reinicie el sistema.

Paso 2 En la máquina de origen (devstack1) cree el directorio cloudera y en la máquina de destino (devstack2) cree el directorio cloudera1.

Paso 3 Guarde final.sh, migration.py y MigrationStrategy.py en cloudera y run.sh en cloudera1 y otorgue los permisos de ejecución para los scripts de shell.

Paso 4 Ejecute final.sh que ejecutará MigrationStrategy.py y en función del valor que devuelve migrate.sh se ejecuta y el control pasa a la VM de destino.

Paso 5 Ejecute run.sh en cloudera1 en el destino seguido de * docker run -it prodready / bin / bash * que ejecutará la imagen del contianer.

Paso 6 Podemos ejecutar las * imágenes docker * en el destino para ver que la ID de la imagen es la misma que la de la máquina virtual host.

ALGORITHM

Paso 1: Hacer que la CPU analice los datos de monitoreo de los recursos en cloud.cpu_util.py obtiene los datos de CPU en tiempo real de los recursos en la nube. Esto irá como una entrada al paso 2.

Paso 2: patternMatching.py: en esto tomamos los datos de monitoreo de la CPU en forma de lista. Este archivo implementa el algoritmo de coincidencia de patrones que realiza el paso de preprocesamiento y coincidencia para obtener la cadena de tiempo de CPU predicha. Esto va como una entrada al paso 3.

Paso 3: trigger_strategy.py: en este paso implementamos la estrategia de activación que logra el mecanismo de elasticidad de computación en la nube con un mínimo retraso de escala de recursos en la nube.

Paso 4: se activa el paso Docker que inicia la migración de contenedores entre máquinas