docker run -p 8080:8080 --memory=4g -p 5005:5005 -v /Users/vnXXX/secrets/vulcan:/secrets -e CATALINA\_OPTS="-Xmx3600M -Druntime.context.system.property.override.enabled=true -Dscm.snapshot.enabled=true -Dscm.root.dir=/tmp/scm -Dis\_keywhiz\_enabled=true -Druntime.context.appName=vulcan-app -Druntime.context.appVersion=1.0 -Druntime.context.environmentType=qa-int -Druntime.context.environment=ca -Druntime.context.cloud=stg-wcnp-cloud -Druntime.context.environment.profile=qa -Dcom.walmart.platform.logging.kafka.brokerList=kafka.kafka-cluster-telemetry.prod1-ndc.walmart.com:9092 -Duser.timezone=UTC -Djavax.net.ssl.trustStore=/app/.certs/rapidssl.jks -Djavax.net.ssl.trustStorePassword=changeit -Djsse.enableSNIExtension=true -Dcom.walmart.platform.logging.console.enabled=true -Dcom.walmart.platform.logging.file.enabled=false -Dcom.walmart.platform.logging.file.format=V4Json -Dcom.walmart.platform.logging.file.path=/dev/null -Dcom.walmart.platform.logging.includeLocation=true -Dcom.walmart.platform.logging.input.level=INFO -Dcom.walmart.platform.logging.kafka.enable=true -Dcom.walmart.platform.logging.logfile.path=/dev/null -Dcom.walmart.platform.logging.network.enabled=true -Dcom.walmart.platform.metrics.file.format=V3JsonMetrics -Dcom.walmart.platform.metrics.file.path=/dev/null -Dcom.walmart.platform.metrics.logfile.path=/dev/null -Dcom.walmart.platform.txnmarking.logfile.path=/dev/stdout -Dcom.sun.management.jmxremote.rmi.port=9090 -Dcom.sun.management.jmxremote=true -Dcom.sun.management.jmxremote.port=9090 -Dcom.sun.management.jmxremote.ssl=false -Dcom.sun.management.jmxremote.authenticate=false -Dcom.sun.management.jmxremote.local.only=false -Djava.rmi.server.hostname=localhost -agentlib:jdwp=transport=dt\_socket,server=y,suspend=n,address=5005" vulcan-app-ca

This Docker command runs a containerized instance of the vulcan-app-ca application with several configurations related to memory limits, port mapping, environment variables, and Java options. Let’s break it down:

### **1. docker run**

* This is the base command to start a new Docker container.

### **2. -p 8080:8080**

* This maps port 8080 on the host machine to port 8080 inside the container. This is typically used for exposing web services running inside the container to the outside world (e.g., accessing a web server).

### **3. --memory=4g**

* This limits the container to use a maximum of 4 GB of memory. This helps control the resource usage of the container.

### **4. -p 5005:5005**

* This maps port 5005 on the host machine to port 5005 in the container. Port 5005 is often used for remote debugging, particularly with Java applications.

### **5. -v /Users/vn53myk/secrets/vulcan:/secrets**

* This mounts a volume from the host machine (at /Users/vn53myk/secrets/vulcan) into the container's /secrets directory. It allows the container to access secrets, configuration files, or certificates that are stored on the host.

### **6. -e CATALINA\_OPTS=...**

* This passes the CATALINA\_OPTS environment variable to the container, which contains a series of Java Virtual Machine (JVM) options. These options configure how the JVM will run the vulcan-app-ca application.

Let’s break down the options passed via CATALINA\_OPTS:

* **-Xmx3600M**: Limits the maximum heap size of the JVM to 3600 MB.
* **-Druntime.context.system.property.override.enabled=true**: Enables overriding of system properties.
* **-Dscm.snapshot.enabled=true**: Enables the snapshotting feature for the Source Code Management (SCM) system.
* **-Dscm.root.dir=/tmp/scm**: Sets the root directory for SCM files to /tmp/scm.
* **-Dis\_keywhiz\_enabled=true**: Indicates that the Keywhiz secrets management system is enabled.
* **-Druntime.context.appName=vulcan-app**: Defines the application name as vulcan-app.
* **-Druntime.context.appVersion=1.0**: Sets the application version to 1.0.
* **-Druntime.context.environmentType=qa-int**: Specifies that the environment type is qa-int, likely for quality assurance purposes.
* **-Druntime.context.environment=ca**: Specifies that the environment is ca (likely short for Canada).
* **-Druntime.context.cloud=stg-wcnp-cloud**: Specifies the cloud environment as stg-wcnp-cloud (a staging environment).
* **-Druntime.context.environment.profile=qa**: Sets the environment profile to qa, which is commonly used for testing.
* **Kafka Logging Options**:
  + **-Dcom.walmart.platform.logging.kafka.brokerList=...**: Specifies the Kafka broker for logging purposes, allowing logs to be sent to a remote Kafka server.
  + **-Dcom.walmart.platform.logging.kafka.enable=true**: Enables Kafka-based logging.
* **SSL Configuration**:
  + **-Djavax.net.ssl.trustStore=/app/.certs/rapidssl.jks**: Specifies the location of the truststore (a file containing trusted certificates) for SSL connections.
  + **-Djavax.net.ssl.trustStorePassword=changeit**: Defines the password for the truststore.
* **Logging Configuration**:
  + Several options (-Dcom.walmart.platform.logging.console.enabled=true, -Dcom.walmart.platform.logging.file.enabled=false, etc.) configure where logs should be sent (e.g., console, files, etc.) and their format.
  + **-Dcom.walmart.platform.logging.logfile.path=/dev/null**: Redirects log output to /dev/null, effectively discarding it.
* **JMX Remote Management**:
  + **-Dcom.sun.management.jmxremote.\***: These options enable JMX (Java Management Extensions) remote monitoring and management via RMI (Remote Method Invocation). This allows administrators or developers to connect to the JVM running inside the Docker container to monitor or manage it.
  + **-Dcom.sun.management.jmxremote.port=9090**: Specifies the port for remote JMX connections.
  + **-Dcom.sun.management.jmxremote.ssl=false**: Disables SSL for JMX remote connections.
  + **-Dcom.sun.management.jmxremote.authenticate=false**: Disables authentication for JMX connections (which might be risky in production environments).
  + **-Djava.rmi.server.hostname=localhost**: Binds the RMI server to localhost.
* **Java Debugging**:
  + **-agentlib:jdwp=transport=dt\_socket,server=y,suspend=n,address=5005**: Enables remote debugging on port 5005 using the Java Debug Wire Protocol (JDWP). This allows you to connect a debugger to the running application.

### **7. vulcan-app-ca**

* This is the name of the Docker image being run. The image likely contains the vulcan-app-ca application, which is started when the container is created.

### **Summary**

This Docker command:

* Runs a Java-based application (vulcan-app-ca) inside a container.
* Limits memory usage to 4 GB.
* Exposes ports 8080 and 5005 (for web service and debugging).
* Mounts a secrets directory from the host machine.
* Passes various Java options to configure memory, logging, environment, and SSL settings.
* Enables remote monitoring and debugging via JMX and JDWP.

This is typically used for a staging environment (qa-int), with logs sent to Kafka, debugging enabled, and secrets mounted from the host machine.