Automating Daily Stock Market Digest Emails Using Docker, Kubernetes, and Azure

Comprehensive Project Documentation

This document provides a detailed account of the steps undertaken, challenges encountered, and the roles Docker and Kubernetes played in implementing the project. It also explores the potential future applications of these technologies.

# Project Objective

The objective was to develop an automated system to send daily stock market digest emails, which included:

* Stock market data with technical indicators
* Top business news headlines

The solution was designed to:

* Be containerized for portability and consistency using Docker
* Automate execution using Kubernetes CronJobs
* Deploy on Azure Kubernetes Service (AKS) for scalability and reliability

# Steps Followed

## Step 1: Writing the Python Script

Tasks Completed:

We developed a Python script (`stockdata.py`) that:

* Fetches stock data using `yfinance`
* Retrieves top business news using the News API
* Sends an email using Yahoo Mail’s SMTP server

Challenges Encountered and Solutions:

* Stock Data Retrieval:
* Some stocks returned empty data due to API limitations
* Solution: Implemented error handling to skip stocks with insufficient data
* Email Sending:
* Yahoo Mail required app-specific passwords for SMTP authentication
* Solution: Enabled app passwords in Yahoo Mail settings
* Scheduling Execution Locally:
* Used the `schedule` library to run the script daily at 7:30 AM

## Step 2: Containerizing the Script with Docker

Tasks Completed:

We used Docker to encapsulate the script along with its dependencies into a portable container.

Steps:

* Created a `Dockerfile`:

*FROM python:3.9-slim*

*WORKDIR /app*

*COPY requirements.txt .*

*RUN pip install --no-cache-dir -r requirements.txt*

*COPY stockdata.py .*

*CMD ["python", "stockdata.py"]*

* Built the Docker image:

*docker build -t python-email-scheduler .*

* Tested the container locally:

*docker run --rm python-email-scheduler*

**Challenges Encountered and Solutions:**

* Dependency Issues:
  + Accidentally included libraries (`smtplib`, `email.mime`) in `requirements.txt`
  + Solution: Removed them as they are part of Python’s standard library
* Environment Consistency:
  + Ensured consistent execution across different systems by isolating dependencies within the container

## Step 3: Scheduling with Kubernetes CronJobs

Tasks Completed:

We utilized Kubernetes CronJobs to automate running the script daily at 7:29 AM UTC.

Steps:

* Created a `cronjob.yaml` file:

*apiVersion: batch/v1*

*kind: CronJob*

*metadata:*

*name: email-sender-cronjob*

*spec:*

*schedule: "29 7 \* \* \*"*

*jobTemplate:*

*spec:*

*template:*

*spec:*

*containers:*

*- name: email-sender-container*

*image: .azurecr.io/python-email-scheduler:latest*

*restartPolicy: OnFailure*

*successfulJobsHistoryLimit: 1*

*failedJobsHistoryLimit: 3*

* Deployed the CronJob using `kubectl`:

*kubectl apply -f cronjob.yaml*

* Verified deployment and monitored logs:

*kubectl get cronjob*

*kubectl logs*

**Challenges Encountered and Solutions:**

* Timezone Mismatch:
  + Kubernetes schedules jobs in UTC, requiring adjustments for local time zones
  + Solution: Converted local time to UTC for scheduling
* Testing Without Waiting for Schedule:
  + Manually triggered jobs using kubectl create job

## Step 4: Deployment on Azure Kubernetes Service (AKS)

Tasks Completed:

We deployed the solution on AKS to ensure scalability and reliability.

Steps:

* Created an AKS cluster using Azure CLI:

*az aks create --resource-group --name --node-count 1 --generate-ssh-keys*

* Pushed the Docker image to Azure Container Registry (ACR):

*docker tag python-email-scheduler .azurecr.io/python-email-scheduler:latest*

*docker push .azurecr.io/python-email-scheduler:latest*

* Connected AKS to ACR and deployed the CronJob YAML file:

*az aks update --name --resource-group --attach-acr*

*kubectl apply -f cronjob.yaml*

* Monitored execution using `kubectl` commands

**Challenges Encountered and Solutions:**

* Subscription Not Registered for AKS:
  + Encountered an error during AKS creation due to missing resource provider registration
  + Solution: Registered `Microsoft.ContainerService` using Azure CLI
* Image Pull Errors:
  + Pods failed to pull images from ACR due to missing permissions
  + Solution: Attached ACR to AKS cluster using `az aks update`

# How Docker Helped

Docker provided significant advantages during development and deployment:

**Benefits:**

1. Environment Isolation:
   * Ensured consistent execution across different systems by isolating dependencies within containers
2. Portability:

* Containers can run on any platform (local, cloud-based) without modification

1. Simplified Dependency Management:

* Resolved version conflicts by packaging dependencies within containers

1. Reusability:

* The same container image can be reused across development, testing, and production environments

1. Future Use Cases:

* Docker can be used for scaling applications, integrating microservices architecture, or deploying complex workflows seamlessly across cloud platforms

# How Kubernetes Helped

Kubernetes provided robust automation and scalability for running scheduled tasks:

**Benefits:**

1. Scheduled Task Automation:
   * Automated running of tasks at specific intervals without manual intervention (e.g., daily stock email)
2. Resource Efficiency:

* Optimized resource usage by scheduling jobs during off-peak hours

1. Fault Tolerance:

* Automatically retried failed jobs, ensuring reliability in execution

1. Scalability:

* Dynamically scaled resources based on workload demands, making it suitable for future large-scale deployments

1. Future Use Cases:

* Kubernetes can be leveraged for more complex workflows, such as multi-step batch processing or integrating machine learning pipelines

# Outcome

The project successfully automated the sending of daily stock market digest emails using Docker, Kubernetes CronJobs, and Azure Kubernetes Service (AKS). Both Docker and Kubernetes were instrumental in ensuring the solution’s portability, scalability, and reliability.