UDACITY

< Return to Classroom

DISCUSS ON STUDENT HUB >

TechTrends

REVIEW

CODE REVIEW 6

HISTORY

Meets Specifications

Good job! You've done amazing work on the project with standout suggestions.

Answering your questions, You'll have to check with the Udacity support channel to decide to make them public or just try posting in the Knowledge forum so that someone could come up with a better answer.

Thank you.

Apply Best Practices For Application Deployment

✓

Create a **"/healthz"** endpoint

- Return a 200 HTTP JSON response with a message *"OK - healthy"*

Return a 200 HTTP JSON response with a message "OK - healthy" ☑

✓

Create a **"/metrics"** endpoint

- Return a 200 HTTP JSON response
- Return the updated amount of posts (*"post_count"*) and number of connections made with the database (*"db_connection_count"*)

✓

Add **logs** to the application

- Record the following events to STDOUT & STDERR:
  - Access an article
  - 404 pages when a non-existing articles are accessed
  - Access "About Us" page
  - A new article is created
- Record Python generic logs at DEBUG level
- Each logline should have a timestamp

Docker for Application Packaging

✓

Create a Dockerfile, including the following instructions:

- Base image: Python in version 2.7
- Expose the application port 3111
- Install packages defined in the requirements.txt file
- Ensure that the database is initialized with the pre-defined posts in the init_db.py file
- The application should execute on the container start

Place the Dockerfile in the project directory

✓

Build the Docker image

- Use Docker commands to build the TechTrends Docker image:
  - Use the tag option to the image name as being "techtrends"
  - Specify the location of the Dockerfile
- Place the used Docker command in the "docker_commands" file

✓

Test the Docker image

- Use Docker commands to execute the Docker image locally, with the following specifications:
  - Using the detached mode

    - Exposed on port “7111” on the host
- Application should be accessible on the “http://127.0.0.1:7111” endpoint.
- Place a screenshot of the output, including the navigation bar, in the "screenshots" folder with the name *“docker-run-local”*.
- Place the used Docker commands to execute the image and retrieve the logs in the "docker_commands" file.
- Copy and paste the Docker container logs in the "docker_commands" file.

Continuous Integration with GitHub Actions

✓

Create a GitHub Action to build and push the application:

- Trigger on every push to the *main* branch
- Executed on the *“ubuntu-latest”* operating system
- *(build and push step)* Context should be set to the project directory
- *(build and push step)* Reference the location for TechTrends Dockerfile
- *(build and push step)* Image pushed to DockerHub with the tag *“techtrends:latest”*
- Place the screenshot with a successful build of the Docker build and push GitHub action, in the "screenshots" folder with the name *“ci-github-actions”.*
- Place the screenshot of the DockerHub TechTrends image, with the tag *“latest”*, in the "screenshots" folder with the name *“ci-dockerhub”*

Kubernetes Declarative Manifest

✓

Deploy a Kubernetes cluster using k3s

- Use a vagrant box to deploy the cluster
- Use k3s to deploy the cluster
- Place the screenshot of the *“kubectl get no”* output, and place it in the "screenshots" folder with the name “k8s-nodes”.

✓

Construct Kubernetes declarative manifests to deploy an application

- Place the “namespace.yaml, deploy.yaml and service.yaml” in the "kubernetes" project folder.
- Namespace with the name “sandbox”
- Deployment with the name “techtrends” in the sandbox namespace running:
    - Image “techtrends:latest”
    - Replicas: 1
    - Resources requests of CPU 250m and memory 64Mi; and limits of CPU 500m and memory

128Mi
    - Container port: “3111”
    - Liveness and readiness probes checking on “/healthz” path on port “3111”
- Service with the name “techtrends” in the sandbox namespace with:
    - Exposed port should be“4111” and target port “3111”
    - Protocol set to “TCP”
    - Service type set to ClusterIP

✓

Apply Kubernetes declarative manifests

- Use kubectl commands to deploy the YAML manifests
- Ensure TechTrends namespace, deployment and service resources are up and running
- Take a screenshot of the *“kubectl get all -n sandbox”* output and place it in the "screenshots" folder with the name “kubernetes-declarative-manifests”.

Helm Charts

✓

Create a Helm Chart

- Stored in the "helm" project folder
- Chart.yaml file should contain:
    - apiVersion: v1
    - name: techtrends
    - version: 1.0.0
- Templates folder should have parameterized namespace, deployment, and service YAML manifests
- Default input values files should contain:
    - Namespace: sandbox
    - ClusterIP service exposing port “4111” and target port “3111” using “TCP” protocol
    - Image “techtrends:latest” with pull policy set to “IfNotPresent”
    - Replicas: 1
    - Resources requests of CPU 250m and memory 64Mi; and limits of CPU 500m and memory 128Mi
    - Container port: “3111”
    - Liveness and readiness probes checking on “/healthz” path on port “3111”

✓

Custom input files for multiple environments

- Values files for staging and production environments stored in the "helm" project folder
- The “values-staging.yaml” file should contain:

    - Namespace: staging
    - Service port set to “5111”
    - Replicas: 3
    - Resources requests of CPU 300m and memory 90Mi; and limits of CPU 500m and memory 128Mi

- The “values-prod.yaml” file should contain:
    - Namespace: prod
    - Service port set to “7111”
    - Image pull policy: “Always”
    - Replicas: 5
    - Resources requests of CPU 350m and memory 128Mi; and limits of CPU 500m and memory 256Mi

Continuous Delivery with ArgoCD

✓

Install ArgoCD

- Install ArgoCD using the official guide
- Expose ArgoCD through a NodePort service
- Access ArgoCD in the local browser
- Take a screenshot of the ArgoCD view once logged in, including the navigation bar, and place it in the "screenshot" folder with the name “argocd-ui”

✓

Build ArgoCD Application manifests

- Should use the TechTrends Helm chart
- For staging deployment the ArgoCD application should be stored in “helm-techtrends-staging.yaml” file
    - Name: techtrends-staging
    - Reference values file: values-staging.yaml
- For prod deployment the ArgoCD application should be stored in “helm-techtrends-prod.yaml” file
    - Name: techtrends-prod
    - Reference values file: values-prod.yaml

✓

Deploy TechTrends with ArgoCD

- Use kubectl commands to deploy the ArgoCD Application resources
- Synchronize the ArgoCD applications and ensure you have 2 new namespaces, *staging* and *prod*, a deployment and service for each environment

- Take a screenshot of the “techtrends-staging“ Application, with synchronized resources (all should be up and running), and store it with the “argocd-techtrends-staging” name.
- Take a screenshot of the “techtrends-prod“Application, with synchronized resources (all should be up and running), and store it with the “argocd-techtrends-prod” name.

DOWNLOAD PROJECT

6 CODE REVIEW COMMENTS ›

RETURN TO PATH

Rate this review

START