Onboard K8s service to CI/CD v2.0

Github Actions is a Continuous Integration and Continuous Deployment (CI/CD) tool that provides a powerful and flexible platform for automating software development workflows. With Github Actions, developers can define custom workflows that automatically build, test, and deploy their code whenever changes are made to the repository.

By combining containerisation and Github Actions, developers can create a robust and efficient CICD pipeline for their containerised applications. Containerisation ensures that the application can run consistently across different environments, while Github Actions automates the build, test, and deployment processes. This results in faster and more reliable software releases, with fewer errors and less manual intervention. This document scopes the services that **are containerised** and **needs to be deployed on EKS**.

> 💡 In this version of CICD we have created a golden helm chart that can be used to deploy any service on k8s end to end including ingress which was not supported in Old CICD setup.

To Onboard a new service Containerised service in Github Actions CICD follow these steps:

Prerequisites:

- DockerFile must be present in the branch you want to deploy on EKS.
- Discuss specific hardware requirements-if any(GPU/Inference) for running the particular service with DevSecOps first.
- If there is a secret is used by the application like DB username/passwords, get them added as SSM parameters by opening a request here.
- Get Github Actions enabled on the Github repo(if not enabled already) you want to deploy with the help of DevSecOps.

Steps:

1. Create a helm-chart values file for the service. Helm is a package manager for Kubernetes and makes it easier to define configs for a service to deploy in Kubernetes Cluster
    a. Create a new branch from master of aws-code repo.
    b. Create helm values file with format `aws-code/helm/<service-name>/<region>/<environment>.yaml` using this file as template.

       For example: if you are deploying a service named **bubble** in **us-east-2** region in **dev** environment then create the file as `aws-code/helm/bubble/us-east-2/dev.yaml`
    c. Fill in the file created with inputs specific to your k8s service like deployment, service, ingress details.

       Note: Everything is configurable in this including deployment, service, ingress, pv/pvc, secrets, hpa, etc in this file if you don't want to use a specific feature then you can leave it commented as it is.
    d. Create a pull request to master branch on aws-code and get it reviewed and merged by Leads and DevSecOps.
2. Add below file k8s-cicdv2.yaml in path `.github/workflows/k8s-cicdv2.yaml` in the service repo's default branch and the branch that you want to deploy.

▼ k8s-cicdv2.yaml

```
name: Build and Deploy to K8s CICD v2.0
on:
  workflow_dispatch:
    inputs:
      service:
        description: 'Service Name'
        type: string
        required: true
        default: '<service-name>'
      region:
        description: 'Region'
        type: choice
        required: true
        default: 'us-east-2'
```

```
        options:
        - us-east-2
        - ca-central-1
        - eu-west-2
      environment:
        description: 'environment'
        type: choice
        required: true
        default: 'dev'
        options:
        - dev
        - qa
        - staging
        - prod
        - pci
      freeze:
        description: 'Freeze requirement.txt for Prod?'
        type: choice
        required: true
        default: false
        options:
        - true
        - false
      namespace:
        description: 'K8s namespace you want deploy this s
        type: string
        required: true
        default: '<namespace of service>'
      deploy_path:
        description: 'Repo path to deploy'
        type: string
        required: true
        default: '/'
jobs:
  Build-Deploy:
    runs-on: ${{ github.event.inputs.environment }}
    permissions:
      id-token: write
```

```
      contents: read
    steps:
      - name: Cleanup build folder
        run: |
          ls -la ./
          rm -rf ./* || true
          rm -rf ./.??* || true
          ls -la ./
      - name: Checkout code
        uses: actions/checkout@v3
        with:
          path: ./repo
      - name: Checkout Helm Chart repository
        uses: actions/checkout@v3
        with:
          repository: prodigal-tech/aws-code
          path: ./aws-code
          token: ${{ secrets.GLOBAL_GITHUB_HTTPS_TOKEN }}
      - name: Build and Deploy
        env:
          service: ${{ inputs.service }}
          environment: ${{ inputs.environment }}
          deploy_path: ${{ inputs.deploy_path }}
          region: ${{ inputs.region }}
          freeze: ${{ inputs.freeze }}
          namespace: ${{ inputs.namespace }}
          branch: ${{ github.ref_name }}
          home: ${{ github.workspace }}
          sha: ${{ github.sha }}
          repo: ${{ github.repository }}
          pat: ${{ secrets.CICD_REQUIREMENT_FREEZE_PAT }}
        run: |
          chmod +x aws-code/cicdv2.sh
          aws-code/cicdv2.sh $region $service $branch $env
```

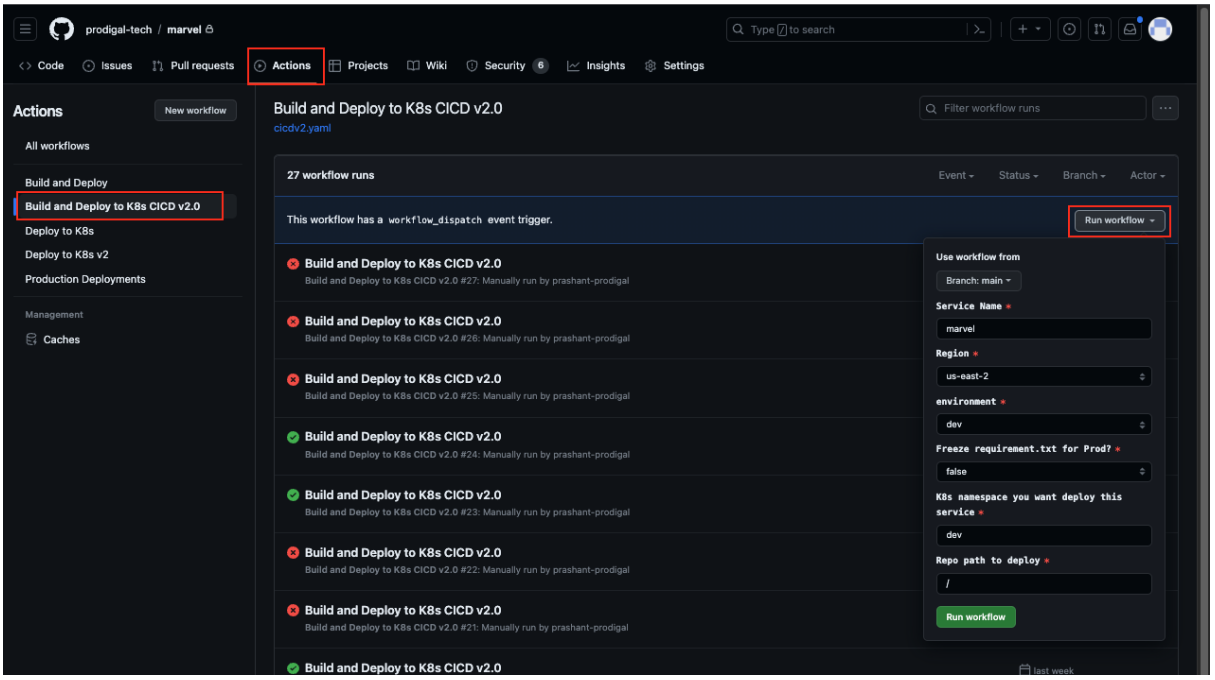
Add below values based on your repo/service:

`<service-name>` : name of service you want to deploy

`Freeze` : Set to true if you want to freeze requirements.txt and push it to main branch for prod (only relevant for platform services)

`<namespace of service>` : namespace k8s service e.g. dev/prod/matsya etc.

3. Now when you want to deploy the service on EKS, run the pipeline, go to Actions Tab in the repo and Click on `Deploy to K8s` in the left section and check on `Run workflow` , change the fields if required and click the `Run Workflow` button at the bottom.



Fields description:

a. `Service Name`: Name of k8s service e.g. pronotes-py-server, matsya, etc.

b. `Region`: The AWS region you want to deploy the service.

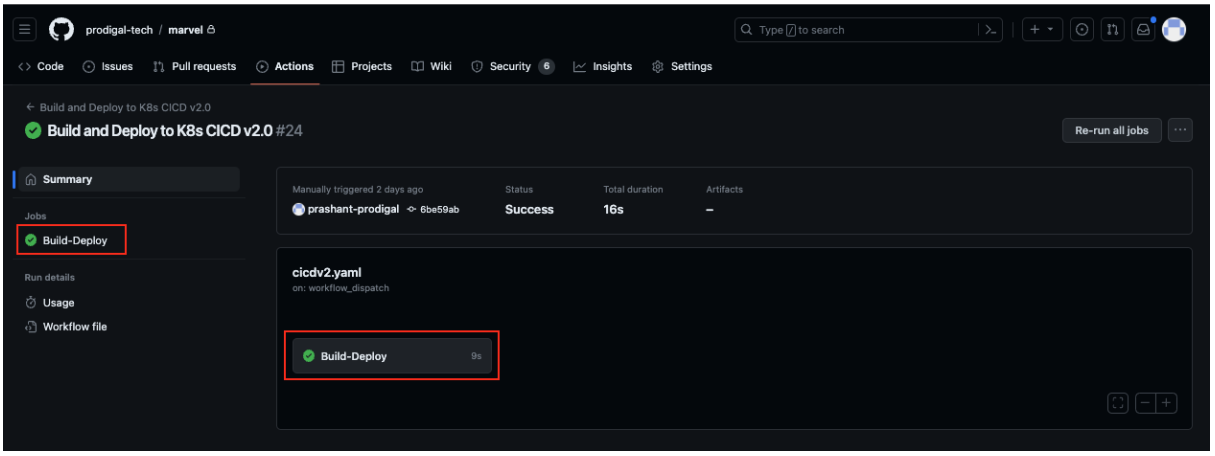
c. `Environment` : the environment you want to deploy this service, i.e. dev, qa, staging or prod.

d. `Freeze requirement.txt for prod?` : Set to true if you want to freeze requirements.txt and push it to main branch for prod (**only relevant for platform python services**). You can leave it false for apps/dsml services

e. `K8s namespace you want to deploy`: Namespace k8s service e.g. dev/prod/matsya etc.

f. `Repo path to deploy`: Path of DockerFile in the repo. If it is on root of the repo then leave this default to `/`

4. You can see the logs of pipeline run on Github Actions by clicking on the run and then on the step



5. This pipeline will deploy everything including deployment, service, ingress, etc.
5. You can check if the pods are running or debug the logs using slack channels `prokube` and `prokube-dev`

Below is the example of a command run in `prokube-dev` channel