Trabalho Final da Disciplina

**Administração de uma PaaS usando Red Hat OpenShift**

Prof.: Paulo César Vigne Júnior

Aluno: Alexandre Nunes Martins

**1.Explique o conceito de PaaS e como plataformas baseadas em kubernetes como o Openshift e Rancher podem ajudar na implementação de ambientes de Contêineres.**

**PaaS**, sigla que significa “Plataforma como Serviço” (do inglês**, P**latform **a**s **a** **S**ervice). É um modelo de serviço em nuvem que disponibiliza uma camada intermediária de desenvolvimento, implantação e gerenciamento de aplicativos que inclui recursos de computação, armazenamento, bancos de dados, integração e ferramentas de desenvolvimento entre a infraestrutura de hardware e a aplicação em si, sem a necessidade de investir em infraestrutura de TI.

O PaaS garante ao usuário o direito de utilizar um serviço personalizado e fácil de configurar. Ele pode lançar rapidamente novos aplicativos e serviços na nuvem, instalar e gerenciar aplicativos próprios ou de terceiros e fazer algumas alterações no software para deixá-lo em conformidade com suas necessidades, permitindo aos desenvolvedores se concentrem no desenvolvimento de aplicativos de alta qualidade sem se preocuparem com a infraestrutura subjacente. O provedor de serviços em nuvem é responsável pela manutenção, atualização e segurança da plataforma. O PaaS também oferece flexibilidade e escalabilidade, permitindo que os usuários ajustem a capacidade de seus aplicativos para atender às demandas do negócio.

O OpenShift é uma plataforma de PaaS que usa o Kubernetes como seu sistema de orquestração de contêineres, permite a criação, implantação e gerenciamento de aplicativos em contêineres de maneira eficiente e escalável. Ele fornece uma estrutura de contêineres para hospedar aplicativos em ambiente de produção, além de permitir a automação de várias tarefas de gerenciamento, como escalabilidade, balanceamento de carga e monitoramento. O OpenShift ajuda na implementação de ambientes de contêineres, fornecendo as seguintes funcionalidades:

- Orquestração de Contêineres com implantação, gerenciamento e escalabilidade de maneira eficiente e automatizada.

- Builds Automatizados com funcionalidade de criação automatizada de imagens de contêineres para aplicativos. Pode detectar alterações no código-fonte e iniciar a construção e a implantação de novas imagens de contêineres sem intervenção manual do usuário.

- Implantação Contínua através de ferramentas para permitir a implantação automatizada de aplicativos em contêineres. Como por exemplo integração com o Jenkins para a criação de pipelines de implantação contínua.

- Gerenciamento Centralizado através de uma única interface centralizada para gerenciamento de contêineres, permite que os usuários gerenciem os recursos do contêiner, como CPU, memória, armazenamento e rede.

- Multi-Tenancy, o OpenShift oferece suporte para implantação de contêineres em vários provedores ou ambientes de usuário, onde vários usuários podem implantar seus aplicativos em contêineres no mesmo cluster de OpenShift sem interferir uns com os outros.

O Rancher é uma plataforma de gerenciamento de contêineres que ajuda na implementação de ambientes de contêineres de maneira fácil e eficiente. Ele fornece uma interface amigável para gerenciamento de contêineres baseados em Docker, Kubernetes e outros orquestradores de contêineres. O Rancher ajuda na implementação de ambientes de contêineres das seguintes maneiras:

- Com implantação Multi-Cluster ele permite a implantação de contêineres em vários clusters de orquestradores de contêineres, permitindo que os usuários gerenciem seus aplicativos em um ambiente heterogêneo. Ele fornece uma única interface para gerenciar clusters Kubernetes, Swarm, Apache Mesos e outros.

- Oferece suporte em provisionamento de infraestrutura para a implantação de contêineres. Ele pode provisionar automaticamente instâncias de máquinas virtuais, hosts físicos e outros recursos para suportar a execução de contêineres em escala.

- Para monitoramento e “logs” o Rancher fornece ferramentas integradas para monitorar e registrar eventos em contêineres e clusters. Ele permite que os usuários monitorem o desempenho, a utilização de recursos e a saúde de seus aplicativos em contêineres, bem como visualizem logs para solução de problemas.

- Permite a criação de pipelines de integração contínua e implantação contínua (CI/CD) para automatizar a implantação de aplicativos em contêineres. Ele pode, por exemplo, se integrar a ferramentas como Jenkins para criar pipelines de CI/CD.

- Oferece várias ferramentas de segurança, como autenticação de usuário, gerenciamento de acesso e políticas de segurança. Ele permite que os usuários configurem políticas de segurança para proteger seus aplicativos e dados.

**2.Explique o conceito de GitOps e como o OpenShift ajudou sua implementação. Crie um desenho explicando o fluxo do GitOps.**

GitOps é um conjunto de práticas definido por um modelo de operações de TI que se concentra em utilizar o Git como a única fonte de informações para declarar e gerenciar a infraestrutura e aplicativos de uma organização. Essa abordagem é baseada no controle de versão, automação, um fluxo de trabalho de integração contínua e implantação contínua (CI/CD), onde todas as alterações no código e infraestrutura são rastreadas e gerenciadas em um repositório Git. Todas as alterações na infraestrutura ou aplicativos são feitas no repositório, e não diretamente nos servidores ou ambientes de produção. Isso permite que as alterações sejam gerenciadas de forma mais estruturada, com revisões de código, aprovações e controle de versão.

O OpenShift fornece suporte nativo para o GitOps. Eis algumas formas com as quais o OpenShift ajuda na implementação do GitOps:

- O OpenShift é integrado com o Git, permitindo que as equipes utilizem o repositório Git como a fonte única da verdade para gerenciar seus aplicativos e infraestrutura. Isso significa que todas as alterações feitas no código e na infraestrutura são registradas e gerenciadas no Git, proporcionando maior controle e visibilidade sobre as alterações.

- OpenShift pode gerenciar a implantação de aplicativos e infraestrutura, permitindo que as equipes automatizem o processo de implantação usando o GitOps. Ferramentas que incluem o Operator Framework, o Tekton Pipelines e o Argo CD, podem ser usadas para definir pipelines de CI/CD e gerenciar as alterações no Git.

- O OpenShift é integrado com outros serviços de DevOps, como por exemplo o Jenkins, permitindo que as equipes utilizem esses serviços em conjunto com o GitOps. Isso possibilita que as equipes criem pipelines de CI/CD com o Jenkins e gerenciem as alterações no Git usando o OpenShift.

- O OpenShift fornece recursos de auditoria e controle para monitorar e gerenciar as alterações feitas no Git, garantindo que as alterações sejam aprovadas e rastreadas antes de serem implementadas em produção.

- O OpenShift oferece a gestão de configuração, que permite gerenciar a configuração da infraestrutura e dos aplicativos usando o Git. Isso significa que todas as configurações são gerenciadas e rastreadas no Git, proporcionando maior controle e visibilidade sobre as alterações de configuração.

**3.Explique o conceito de Pipeline e como o OpenShift ajudou sua implementação. Crie um desenho esboçando um pipeline desde o código no repositório git até sua disponibilização em contêiner.**

Pipeline é um termo utilizado na área de desenvolvimento de software para se referir a um processo de automatização de compilação, testes e implantação de um software. Um pipeline é um conjunto de etapas sequenciais que um software deve passar desde a fase de desenvolvimento até a implantação em produção.

O objetivo principal de um pipeline é automatizar todo o processo de compilação, teste e implantação do software para que o processo ocorra de forma consistente e previsível em todos os ambientes. Com isso, é possível garantir que o software desenvolvido atenda a todos os requisitos e esteja pronto para ser entregue aos usuários.

Um pipeline é construído usando ferramentas de automação que permitem aos desenvolvedores criarem etapas para compilar o código, executar testes automatizados, gerar artefatos, fazer a implantação em ambientes de testes e, eventualmente, em produção. Cada etapa do pipeline é configurada para rodar automaticamente após a conclusão da etapa anterior, permitindo que todo o processo seja executado sem a necessidade de intervenção humana. Isso torna o processo mais rápido, consistente e confiável.

Uma das principais maneiras pelas quais o OpenShift ajuda na implementação de pipelines é fornecendo um recurso chamado "BuildConfig", que permite a criação de pipelines de compilação e implantação de aplicativos em contêineres. O BuildConfig é uma definição de pipeline que especifica as etapas necessárias para compilar, testar e implantar um aplicativo em contêineres, incluindo a construção da imagem do contêiner, a execução de testes automatizados e a implantação em um ambiente de teste ou produção.

Outra maneira pela qual o OpenShift ajuda na implementação de pipelines é fornecendo um ambiente de contêineres seguro e escalável para a execução dos pipelines. O OpenShift permite que as equipes de desenvolvimento criem pipelines em um ambiente isolado, com recursos limitados e controle de acesso restrito, garantindo que os pipelines não afetem outros aplicativos em execução no mesmo cluster. Além disso, o OpenShift oferece suporte para integração contínua e entrega contínua (CI/CD), permitindo que as equipes de desenvolvimento implementem pipelines automatizados que executam a compilação, testes e implantação de aplicativos em contêineres automaticamente.

**4.O que são os OpenShift Operators e como se diferenciam do HelmChart?**

OpenShift Operators é uma extensão da plataforma OpenShift que fornece uma abordagem mais avançada para o gerenciamento de aplicativos em contêineres de maneira automatizada e simplificada. Os OpenShift Operators são usados ​​para gerenciar uma ampla variedade de aplicativos em ambientes de produção. São implementados como contêineres de aplicativos em execução na plataforma OpenShift, que são responsáveis por gerenciar outros contêineres de aplicativos. Eles são escritos em uma linguagem específica para cada Operator, que descreve as operações que o Operator deve executar para gerenciar o aplicativo.

Os OpenShift Operators usam o conceito de "operações declarativas", que permite que os usuários especifiquem o estado desejado de um aplicativo em vez de instruções precisas para alcançar esse estado. Isso significa que os usuários especificam o que desejam, e o Operator se encarrega de implementar e manter o estado desejado. Por exemplo, eles permitem que os usuários gerenciem aplicativos em grande escala de maneira mais fácil, permitindo a implantação, atualização e remoção automatizada de aplicativos. Além disso, os Operators fornecem maior consistência e confiabilidade no gerenciamento de aplicativos, eliminando a necessidade de intervenção manual em muitas operações.

O HelmChart é um gerenciador de pacotes para Kubernetes que permite que os usuários instalem, atualizem e gerenciem aplicativos em contêineres com facilidade. Ele usa arquivos YAML para definir as dependências, configurações e recursos necessários para implantar um aplicativo. O HelmChart usa uma abordagem baseada em modelo para definir como um aplicativo deve ser implantado e gerenciado, permitindo que os usuários personalizem o comportamento do aplicativo de acordo com suas necessidades.

Os OpenShift Operators e o HelmChart são duas ferramentas diferentes usadas para gerenciar aplicativos baseados em contêineres em um ambiente Kubernetes.

Embora tanto os OpenShift Operators quanto o HelmChart sejam usados para gerenciar aplicativos em contêineres, os Operators são mais avançados em termos de automação, simplificação e escala. Eles são mais adequados para ambientes de produção e para aplicativos que exigem gerenciamento mais avançado e personalizado. O HelmChart é mais adequado para aplicativos simples e ambientes de desenvolvimento e teste.

**Prática 1:** No Openshift, crie um projeto chamado "multilayer-app" e disponibilize uma aplicação que possua multiplas camadas, por exemplo, uma web e uma de banco de dados ou fila, completando o exemplo, esta aplicação pode ser um wordpress, drupal ou qualquer uma de sua preferência. A Aplicação precisará manter as boas práticas para seus segredos (secrets) bem como suas configurações (configmaps) e ser exposta via route (ingress). Ignore os volumes persistentes desta aplicação.

* 1. **Project:**

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

* 1. **ConfigMaps:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* + 1. **wordpress-cfg-env.yaml:**

kind: ConfigMap

apiVersion: v1

metadata:

  name: wordpress-cfg-env

  namespace: anm-multilayer-app

  uid: b1a46fda-9cff-47be-b80e-6178288b7116

  resourceVersion: '318724061'

  creationTimestamp: '2023-04-14T11:42:35Z'

data:

  WORDPRESS\_BLOG\_NAME: Blog do Alexandre Nunes

  WORDPRESS\_DATABASE\_HOST: mysql

  WORDPRESS\_DATABASE\_NAME: wordpress

  WORDPRESS\_DATABASE\_USER: wordpress

  WORDPRESS\_USERNAME: admin

* + 1. **mysql-cfg-env.yaml:**

kind: ConfigMap

apiVersion: v1

metadata:

  name: mysql-cfg-env

  namespace: anm-multilayer-app

  uid: c24579cb-b42d-4b49-b988-a85d119aea2d

  resourceVersion: '318687421'

  creationTimestamp: '2023-04-14T11:02:13Z'

data:

  MYSQL\_DATABASE: wordpress

  MYSQL\_USER: wordpress

* 1. **Secrets:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

* + 1. **wordpress-secret-env.yaml:**

kind: Secret

apiVersion: v1

metadata:

  name: wordpress-secret-env

  namespace: anm-multilayer-app

  uid: e307d97e-7c1b-4a94-b4fa-4114546db45f

  resourceVersion: '318714822'

  creationTimestamp: '2023-04-14T11:46:10Z'

data:

  WORDPRESS\_DATABASE\_PASSWORD: SW5mbmV0MjAyM0luZm5ldDIwMjM=

  WORDPRESS\_PASSWORD: SW5mbmV0MjAyM0luZm5ldDIwMjM=

type: Opaque

* + 1. **mysql-secret-env.yaml:**

kind: Secret

apiVersion: v1

metadata:

  name: mysql-secret-env

  namespace: anm-multilayer-app

  uid: f726721b-b0ae-4a7f-96a4-dcd4b865045f

  resourceVersion: '318683907'

  creationTimestamp: '2023-04-14T10:56:09Z'

data:

  MYSQL\_PASSWORD: SW5mbmV0MjAyM0luZm5ldDIwMjM=

  MYSQL\_ROOT\_PASSWORD: SW5mbmV0MjAyM0luZm5ldDIwMjM=

type: Opaque

* 1. **StatefulSet:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

* + 1. **mysql-statefulset.yaml:**

kind: StatefulSet

apiVersion: apps/v1

metadata:

  name: mysql

  namespace: anm-multilayer-app

spec:

  replicas: 1

  selector:

    matchLabels:

      app: mysql

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: mysql

    spec:

      containers:

        - name: mysql

          image: 'bitnami/mysql:5.7.41'

          ports:

            - name: mysql

              containerPort: 3306

              protocol: TCP

          envFrom:

            - configMapRef:

                name: mysql-cfg-env

            - secretRef:

                name: mysql-secret-env

          resources: {}

          terminationMessagePath: /dev/termination-log

          terminationMessagePolicy: File

          imagePullPolicy: IfNotPresent

      restartPolicy: Always

      terminationGracePeriodSeconds: 10

      dnsPolicy: ClusterFirst

      securityContext: {}

      schedulerName: default-scheduler

  serviceName: mysql

  podManagementPolicy: OrderedReady

  updateStrategy:

    type: RollingUpdate

    rollingUpdate:

      partition: 0

  revisionHistoryLimit: 10

* 1. **Deployment:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

* + 1. **wordpress-deployment.yaml:**

kind: Deployment

apiVersion: apps/v1

metadata:

  name: wordpress

  namespace: anm-multilayer-app

spec:

  replicas: 3

  selector:

    matchLabels:

      app: wordpress

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: wordpress

    spec:

      containers:

        - name: wordpress

          image: 'bitnami/wordpress:6.2.0'

          ports:

            - containerPort: 80

              protocol: TCP

          envFrom:

            - configMapRef:

                name: wordpress-cfg-env

            - secretRef:

                name: wordpress-secret-env

          resources: {}

          terminationMessagePath: /dev/termination-log

          terminationMessagePolicy: File

          imagePullPolicy: IfNotPresent

      restartPolicy: Always

      terminationGracePeriodSeconds: 30

      dnsPolicy: ClusterFirst

      securityContext: {}

      schedulerName: default-scheduler

  strategy:

    type: RollingUpdate

    rollingUpdate:

      maxUnavailable: 25%

      maxSurge: 25%

  revisionHistoryLimit: 10

  progressDeadlineSeconds: 600

* 1. **Services:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Tabela

Descrição gerada automaticamente

* + 1. **mysql-service.yaml:**

kind: Service

apiVersion: v1

metadata:

  name: mysql

  namespace: anm-multilayer-app

spec:

  clusterIP: None

  ipFamilies:

    - IPv4

  ports:

    - protocol: TCP

      port: 3306

      targetPort: 3306

  internalTrafficPolicy: Cluster

  clusterIPs:

    - None

  type: ClusterIP

  ipFamilyPolicy: SingleStack

  sessionAffinity: None

  selector:

    app: mysql

* + 1. **wordpress-service.yaml:**

kind: Service

apiVersion: v1

metadata:

  name: wordpress

  namespace: anm-multilayer-app

spec:

  clusterIP: 172.30.47.119

  ipFamilies:

    - IPv4

  ports:

    - protocol: TCP

      port: 8080

      targetPort: 8080

  internalTrafficPolicy: Cluster

  clusterIPs:

    - 172.30.47.119

  type: ClusterIP

  ipFamilyPolicy: SingleStack

  sessionAffinity: None

  selector:

    app: wordpress

* 1. **Route:**

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

* + 1. **wordpress-route.yaml:**

kind: Route

apiVersion: route.openshift.io/v1

metadata:

  name: wordpress

  namespace: anm-multilayer-app

  annotations:

    openshift.io/host.generated: 'true'

spec:

  host: wordpress-anm-multilayer-app.apps.eu410.prod.nextcle.com

  path: /

  to:

    kind: Service

    name: wordpress

    weight: 100

  port:

    targetPort: 8080

  wildcardPolicy: None

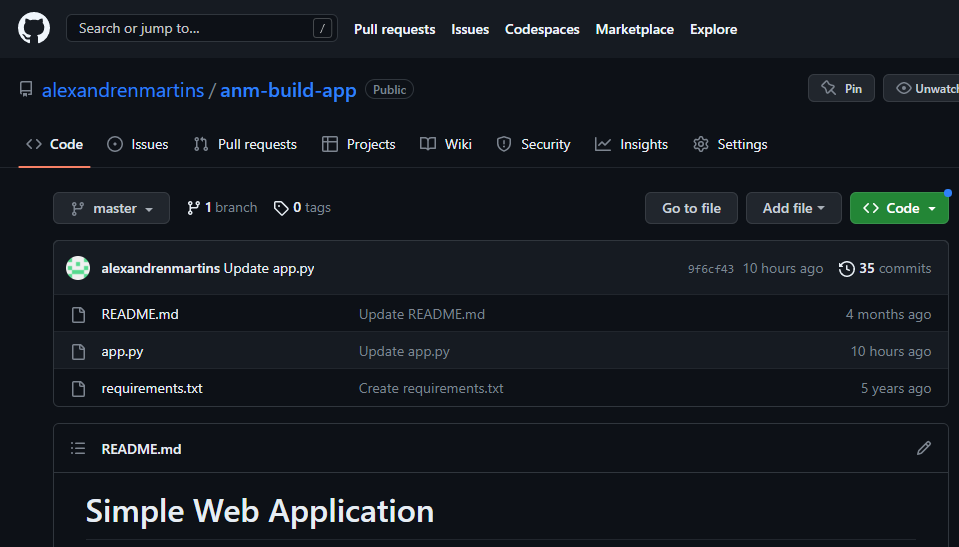
* 1. **Application Group:**

**Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente**

**Prática 2:** No Openshift, crie um projeto chamado "build-app", utilizando o OpenShift Builds realize uma disponibilização de uma aplicação a partir do código fonte oriundo do repositório git. Também deverá ser configurado o Webhook no repositório git para futuros rebuilds. O Resultado final será a aplicação containerizada e seu serviço exposto via route (ingress).

* 1. **Repositório:**



* 1. **Import from Git:**

Tela de celular com aplicativo aberto

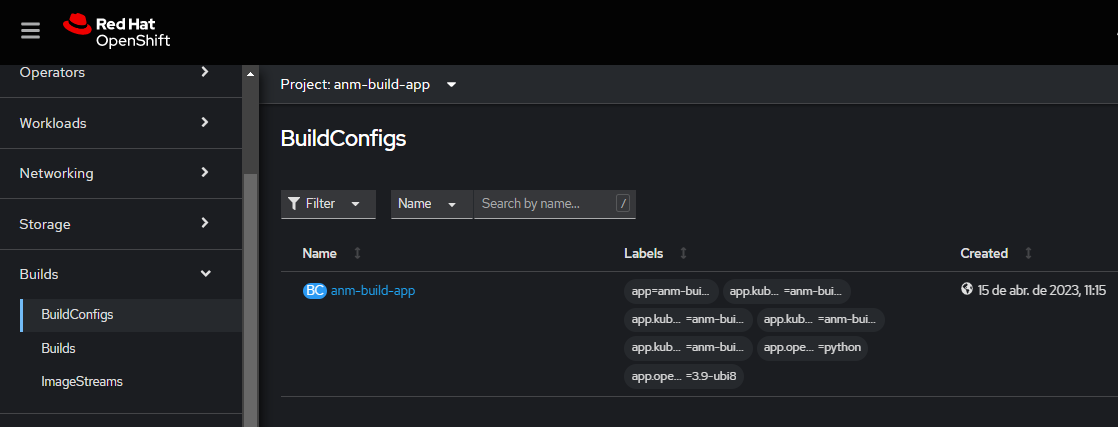
Descrição gerada automaticamente

* 1. **Topology anm-build-app:**

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

* 1. **BuildConfig:**



* 1. **Build:**

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente

* 1. **Pods:**

Tela de computador

Descrição gerada automaticamente

* 1. **Deployment:**

Tela de computador com fundo preto

Descrição gerada automaticamente

* 1. **build-app-deployment.yaml:**

kind: Deployment

apiVersion: apps/v1

metadata:

  name: anm-build-app

  namespace: anm-build-app

  labels:

    app: anm-build-app

    app.kubernetes.io/component: anm-build-app

    app.kubernetes.io/instance: anm-build-app

    app.kubernetes.io/name: anm-build-app

    app.kubernetes.io/part-of: anm-build-app

    app.openshift.io/runtime: python

    app.openshift.io/runtime-version: 3.9-ubi8

spec:

  replicas: 1

  selector:

    matchLabels:

      app: anm-build-app

  template:

    metadata:

      creationTimestamp: null

      labels:

        app: anm-build-app

        deployment: anm-build-app

    spec:

      containers:

        - name: anm-build-app

          image: >-

            image-registry.openshift-image-registry.svc:5000/anm-build-app/anm-build-app@sha256:c3a9365c7f0d4778f53e2012613a503210b523447bb7693ac0041ce7c49fac69

          ports:

            - containerPort: 8080

              protocol: TCP

          terminationMessagePath: /dev/termination-log

          terminationMessagePolicy: File

          imagePullPolicy: Always

      restartPolicy: Always

      terminationGracePeriodSeconds: 30

      dnsPolicy: ClusterFirst

      schedulerName: default-scheduler

  strategy:

    type: RollingUpdate

    rollingUpdate:

      maxUnavailable: 25%

      maxSurge: 25%

  revisionHistoryLimit: 10

  progressDeadlineSeconds: 600

* 1. **Services:**

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

* 1. **buildapp-service.yaml:**

kind: Service

apiVersion: v1

metadata:

  name: anm-build-app

  namespace: anm-build-app

  labels:

    app: anm-build-app

    app.kubernetes.io/component: anm-build-app

    app.kubernetes.io/instance: anm-build-app

    app.kubernetes.io/name: anm-build-app

    app.kubernetes.io/part-of: anm-build-app

    app.openshift.io/runtime: python

  annotations:

    app.openshift.io/vcs-uri: 'https://github.com/alexandrenmartins/anm-build-app.git'

spec:

  clusterIP: 10.217.5.141

  ipFamilies:

    - IPv4

  ports:

    - name: 8080-tcp

      protocol: TCP

      port: 8080

      targetPort: 8080

  internalTrafficPolicy: Cluster

  clusterIPs:

    - 10.217.5.141

  type: ClusterIP

  ipFamilyPolicy: SingleStack

  sessionAffinity: None

  selector:

    app: anm-build-app

    deployment: anm-build-app

* 1. **Route:**

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

* 1. **buildapp-route.yaml:**

kind: Route

apiVersion: route.openshift.io/v1

metadata:

  name: anm-build-app

  namespace: anm-build-app

  labels:

    app: anm-build-app

    app.kubernetes.io/component: anm-build-app

    app.kubernetes.io/instance: anm-build-app

    app.kubernetes.io/name: anm-build-app

    app.kubernetes.io/part-of: anm-build-app

    app.openshift.io/runtime: python

spec:

  host: anm-build-app-anm-build-app.crc-lgph7-master-0.crc.oinmrywi7wst.instruqt.io

  to:

    kind: Service

    name: anm-build-app

    weight: 100

  port:

    targetPort: 8080-tcp

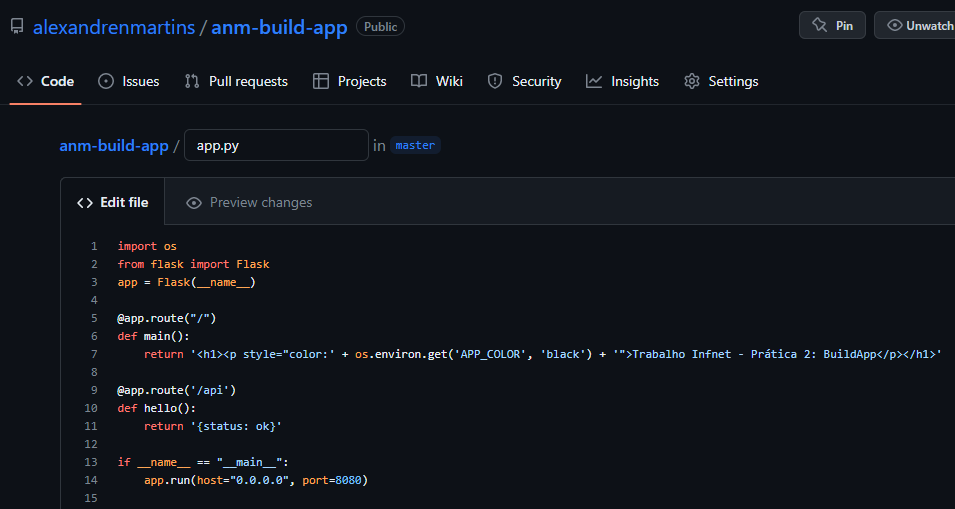
  tls:

    termination: edge

    insecureEdgeTerminationPolicy: Redirect

  wildcardPolicy: None

* 1. **app.py:**



* 1. **BuildApp webpage:**

Texto

Descrição gerada automaticamente com confiança média

* 1. **Webhook configuration:**

Tela de computador com fundo preto

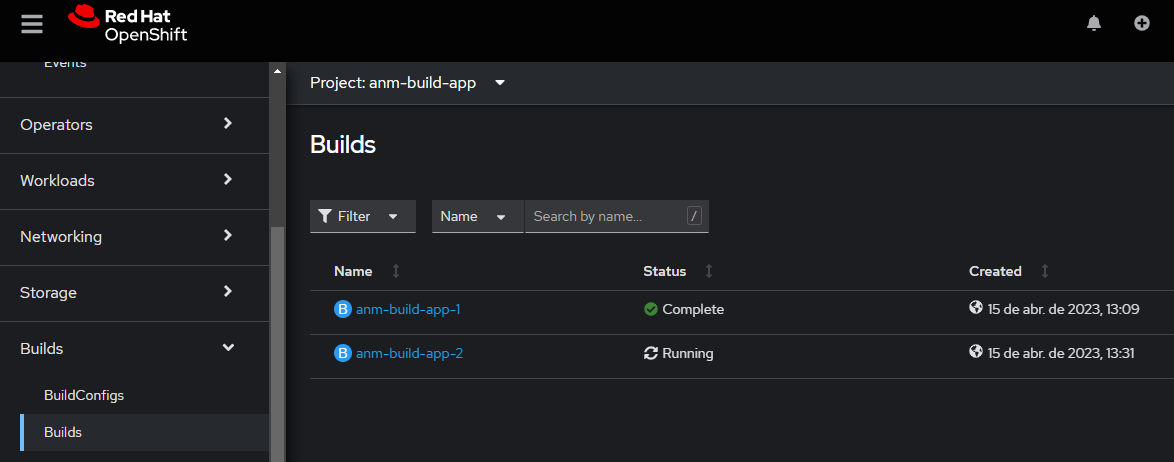
Descrição gerada automaticamente

* 1. **app.py file changed and commited:**

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

* 1. **Build running:**

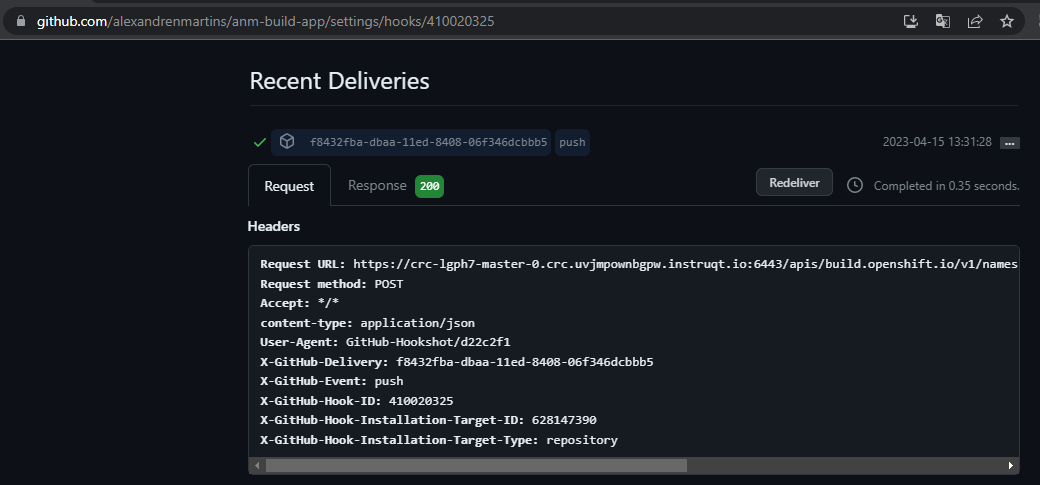


* 1. **Build complete:**

Tela de celular com aplicativo aberto

Descrição gerada automaticamente

* 1. **Webhook Recent Delivery:**



* 1. **BuildApp webpage update:**

Texto

Descrição gerada automaticamente