DEVOPS

ORQUESTRANDO O FUTURO DA ENTREGA DE SOFTWARE

NOVEMBRO/2023

JUCIMAR RODRIGUES

JUCIMAR RODRIGUES

- 08/2005: Início de Tudo
- 02/2009: Ciência da Computação
- 04/2010: Tecnologia da Informação
- 11/2011: Desenvolvimento Desktop
- 10/2014: Desenvolvimento Mobile
- 02/2016: Desenvolvimento Web
- 06/2016: AngelLira
- 01/2021: Squad Leader
- 08/2022: DevOps
- 02/2023: Infraestrutura

TÓPICOS DA **APRESENTAÇÃO**

RESUMO

DEVOPS

KUBERNETES

CI/CD

NOVEMBRO/2023

RESUMO

RESUMO

O que vamos aprender?

- Conceitos sobre o DevOps
- Conceitos sobre o Kubernetes
- Instalar, configurar e operar o Kubernetes
- Instalar, configurar e operar o ArgoCD
- Criar e publicar imagens do Docker
- Criar um projeto automatizado com CI/CD

DEVOPS

DEVOPS

O que é DevOps?

Um composto de Dev (desenvolvimento) e Ops (operações), o DevOps é a união de pessoas, processos e tecnologias para fornecer continuamente valor aos clientes.

DEVOPS

O que o DevOps significa para as equipes?

O DevOps permite que funções anteriormente isoladas – desenvolvimento, operações de TI, engenharia da qualidade e segurança – atuem de forma coordenada e colaborativa para gerar produtos melhores e mais confiáveis.

DEVOPS

O que o DevOps significa para as equipes?

Ao adotar uma cultura de DevOps em conjunto com as práticas e ferramentas de DevOps, as equipes ganham a capacidade de responder melhor às necessidades dos clientes, aumentar a confiança nos aplicativos que constroem e cumprir as metas empresariais mais rapidamente.

DEVOPS

Benefícios do DevOps

Equipes que adotam a cultura, as práticas e as ferramentas de DevOps apresentam alto desempenho, criando produtos melhores, com mais rapidez, para maior satisfação do cliente. Esse aumento na colaboração e na produtividade também é essencial para cumprir metas empresariais como: acelerar a colocação no mercado; adaptar ao mercado e à concorrência; manter a estabilidade e a confiabilidade do sistema; e melhorar o tempo médio de recuperação.

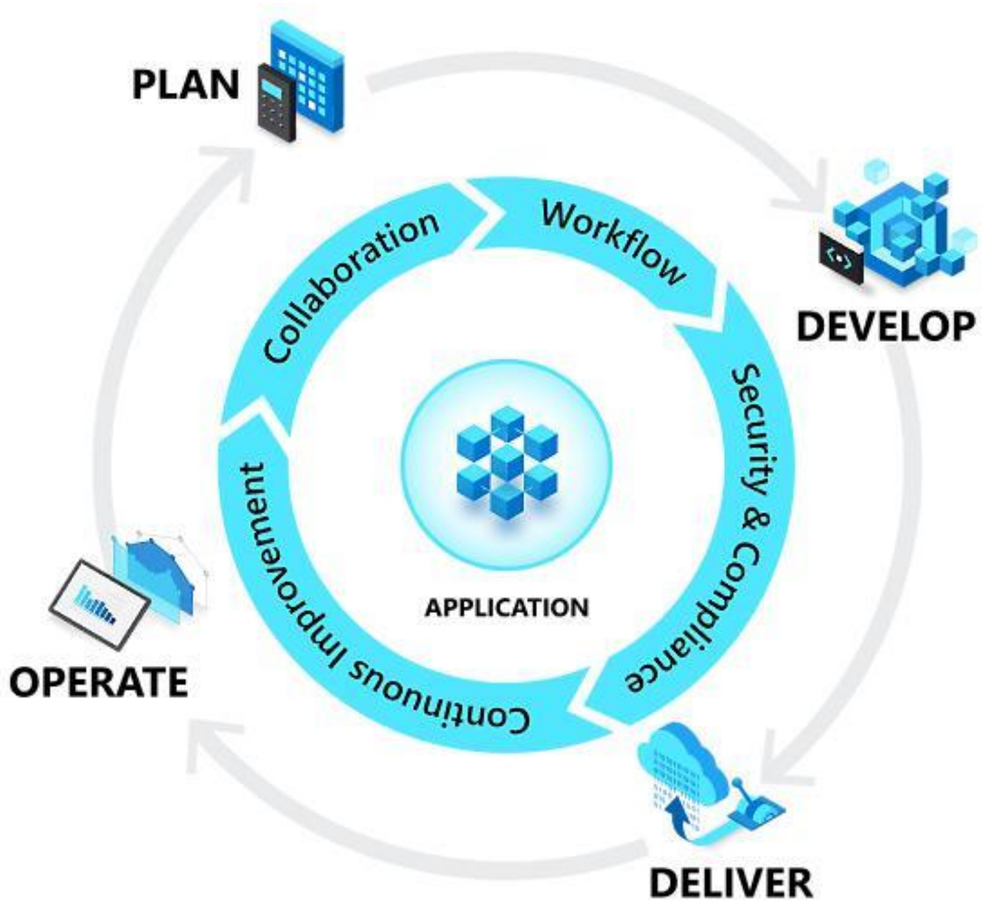
DEVOPS

DevOps e o ciclo de vida do aplicativo

O DevOps influencia o ciclo de vida do aplicativo em todas as fases do **planejamento**, do **desenvolvimento**, da **entrega** e da **operação**. Cada fase depende das demais e elas não são específicas da função. Em uma verdadeira cultura de DevOps, cada função está envolvida de alguma forma em cada fase.

DEVOPS

DevOps e o ciclo de vida do aplicativo



DEVOPS

Ciclo de vida do aplicativo: Planejamento

Na fase de planejamento, as equipes de DevOps idealizam, definem e descrevem os recursos e as funcionalidades dos aplicativos e sistemas que estão construindo. Elas acompanham o progresso em níveis altos e baixos de granularidade, desde tarefas de produto único até tarefas que abrangem portfólios de vários produtos.

DEVOPS

Ciclo de vida do aplicativo: Planejamento

- Criar listas de pendências, acompanhar bugs, gerenciar o desenvolvimento de software Agile com o Scrum, usar quadros Kanban e visualizar o progresso com painéis são algumas das maneiras pelas quais as equipes de DevOps planejam com agilidade e visibilidade.

DEVOPS

Ciclo de vida do aplicativo: Desenvolvimento

- A fase de desenvolvimento inclui todos os aspectos da codificação – gravação, teste, revisão e integração do código pelos membros da equipe – bem como a compilação do código em artefatos de compilação, que podem ser implementados em vários ambientes.

DEVOPS

Ciclo de vida do aplicativo: Desenvolvimento

As equipes de DevOps buscam inovar rapidamente sem sacrificar a qualidade, a estabilidade e a produtividade. Para fazer isso, elas usam ferramentas extremamente produtivas, automatizam etapas elementares e manuais e iteram em pequenos incrementos por meio de testes automatizados e integração contínua.

DEVOPS

Ciclo de vida do aplicativo: Entrega

A entrega é o processo de implantação de aplicativos nos ambientes de produção de maneira consistente e confiável. A fase de entrega também inclui a implantação e a configuração da infraestrutura fundamental totalmente governada que compõe esses ambientes.

DEVOPS

Ciclo de vida do aplicativo: Entrega

Na fase de entrega, as equipes definem um processo de gerenciamento de versão com estágios claros de aprovação manual. Elas também definem portões automatizados que movem os aplicativos entre os estágios, até que sejam disponibilizados aos clientes.

DEVOPS

Ciclo de vida do aplicativo: Entrega

- A automação desses processos os torna escalonáveis, repetíveis e controlados. Dessa forma, as equipes que praticam o DevOps podem frequentemente atuar e entregar com facilidade, confiança e tranquilidade.

DEVOPS

Ciclo de vida do aplicativo: Operação

A fase de operação envolve manter, monitorar e solucionar problemas de aplicativos em ambientes de produção. Ao adotar as práticas de DevOps, as equipes trabalham para garantir a confiabilidade do sistema, a alta disponibilidade e o objetivo de tempo de inatividade igual a zero, reforçando a segurança e a governança.

DEVOPS

Ciclo de vida do aplicativo: Operação

- As equipes de DevOps buscam identificar os problemas antes que eles afetem a experiência do cliente e mitigar os problemas rapidamente quando ocorrem. Manter esse nível de vigilância requer telemetria avançada, alertas acionáveis e visibilidade total sobre os aplicativos e o sistema subjacente.

DEVOPS

Cultura do DevOps

Embora a adoção de práticas de DevOps automatize e otimize processos por meio da tecnologia, tudo começa com a cultura dentro da organização e com as pessoas que fazem parte dela. O desafio de cultivar uma cultura de DevOps exige mudanças profundas na maneira como as pessoas trabalham e colaboram.

DEVOPS

Cultura do DevOps

No entanto, quando se comprometem com esse desafio, as organizações podem criar o ambiente ideal para o desenvolvimento de equipes de alto desempenho, onde alguns benefícios começam a ser observados:

- Colaboração, visibilidade e alinhamento
- Mudanças de escopo e de responsabilidade
- Ciclos de versão mais curtos
- Aprendizado contínuo

DEVOPS

Práticas do DevOps

Além de estabelecer uma cultura de DevOps, as equipes dão vida ao DevOps implementando certas práticas ao longo do ciclo de vida do aplicativo. Algumas dessas práticas ajudam a acelerar, automatizar e melhorar uma fase específica. Outras abrangem várias fases, ajudando as equipes a criar processos contínuos que melhoram a produtividade.

DEVOPS

Práticas do DevOps

São algumas práticas comuns do DevOps:

- CI/CD (Integração Contínua e Entrega Contínua)
- Controle de versão
- Desenvolvimento de software Agile
- Infraestrutura como código
- Gerenciamento de configuração
- Monitoramento contínuo

DEVOPS

O DevOps e a nuvem

A adoção da nuvem transformou totalmente a forma como as equipes criam, implantam e operam aplicativos. Isso somado à adoção do DevOps permitiu às equipes aprimorar as próprias práticas e o atendimento aos clientes.

DEVOPS

O DevOps e a nuvem: Agilidade na nuvem

Com a capacidade de provisionar e configurar rapidamente ambientes de nuvem em várias regiões e com recursos ilimitados, as equipes ganham agilidade na implantação dos aplicativos. Agora, em vez de precisar comprar, configurar e manter servidores físicos, as equipes criam ambientes complexos de nuvem em questão de minutos e encerram esses ambientes quando eles deixam de ser necessários.

DEVOPS

O DevOps e a nuvem: Computação sem servidor

Com a maior parte da sobrecarga de gerenciamento da infraestrutura transferida para o provedor de nuvem, as equipes podem se dedicar aos aplicativos, em vez de se preocupar com a infraestrutura subjacente. A computação sem servidor possibilita executar aplicativos sem precisar configurar e manter servidores. Algumas opções reduzem a complexidade e o risco da implantação e das operações.

DEVOPS

O DevOps e a nuvem: Kubernetes

Com o aumento dos aplicativos que usam a tecnologia de contêineres, o Kubernetes está se tornando a solução preferida do setor para orquestrar contêineres em escala. Automatizar os processos de criação e implantação de contêineres por meio de pipelines de CI/CD e monitorar esses contêineres na produção estão se tornando práticas essenciais na era do Kubernetes.

KUBERNETES

KUBERNETES

O que é Kubernetes?

Os clusters Kubernetes hospedam aplicativos em contêineres de maneira confiável e escalonável. Tendo o DevOps em mente, o Kubernetes simplifica as tarefas de manutenção, como as atualizações, por exemplo.

KUBERNETES

O que é MicroK8s?

MicroK8s é uma implantação upstream do Kubernetes realizada pela Canonical e certificada pela CNCF (Cloud Native Computing Foundation) que é executada inteiramente em sua estação de trabalho ou dispositivo de borda.

KUBERNETES

Requisitos para a Instalação do MicroK8s

KUBERNETES

Instalação do MicroK8s

Tradicional:

- *sudo snap install microk8s --classic*

Escolhendo uma versão específica:

- *sudo snap install microk8s --classic --channel=1.28/stable*

Listando todas as versões disponíveis:

- *snap info microk8s*

Mudando a versão escolhida:

- *sudo snap refresh microk8s --channel=latest/edge*

KUBERNETES

Configuração do MicroK8s

Verificando o status do MicroK8s:

- *sudo microk8s status*

Removendo a necessidade de sudo:

- *sudo usermod -aG microk8s $USER*

Recarregar os grupos de usuários:

- *newgrp microk8s*

Verificando o status do MicroK8s sem sudo:

- *microk8s status*

KUBERNETES

Configuração do kubectl

Testando o comando kubectl:

- *microk8s kubectl get all -A*

Criando um alias para o comando kubectl:

- *sudo nano ~/.bashrc*
- *alias kubectl="microk8s kubectl“*
- *. ~/.bashrc*

Testando o comando kubectl sem o prefixo:

- *kubectl get all -A*

KUBERNETES

Instalação de add-ons do MicroK8s

Habilitando o serviço de Rotas:

- *microk8s enable ingress*

Habilitando o serviço de Resolução de DNS:

- *microk8s enable dns*

Habilitando o serviço de Armazenamento:

- *microk8s enable storage*

Habilitando o serviço de IP Fixo (10.0.1.1):

- *microk8s enable host-access*

KUBERNETES

Kubernetes Dashboard

Habilitando o Kubernetes Dashboard:

- *microk8s enable dashboard*

Alterando o tipo do serviço para NodePort:

- *kubectl -n kube-system edit service kubernetes-dashboard*

Descobrindo a porta do serviço:

- *kubectl -n kube-system get services*

Descobrindo o token de acesso:

- *token=$(kubectl -n kube-system get secret | grep default-token | cut -d " " -f1)*
- *kubectl -n kube-system describe secret $token*


CI/CD

CI/CD

Instalação do Argo CD

Gerando o arquivo do kubeconfig:

- *cp /var/snap/microk8s/current/credentials/client.config ~/.kube/config/client.config*

Criando o namespace do argocd:

- *kubectl create namespace argocd*

Instalando o Argo CD:

- *kubectl apply -n argocd -f https://raw.githubusercontent.com/argoproj/argo-cd/stable/manifests/install.yaml*

CI/CD

Configuração do Argo CD

Alterando o tipo do serviço:

- *kubectl patch svc argocd-server -n argocd -p '{"spec": {"type": "LoadBalancer"}}'*

Descobrindo a porta do serviço:

- *kubectl -n argocd get services*

Descobrindo a senha de acesso (admin):

- *verificar a secret argocd-initial-admin-secret*

CI/CD

Criando um aplicativo no Argo CD

- Após logar, clicar no botão + New App
- Na seção General, informar o seguinte:
  - Em Application Name, informar guestbook
  - Em Project Name, selecionar default
  - Manter a Sync Policy como Manual
  - Para as demais opções manter o padrão

CI/CD

Criando um aplicativo no Argo CD

- Na seção **Source**, informar o seguinte:
  - Em **Repository URL**, informar a URL
    - **https://github.com/argoproj/argocd-example-apps.git**
  - Manter a **Revision** como **HEAD**
  - Em **Path**, informar **guestbook**
  - Para as demais opções manter o padrão

CI/CD

Criando um aplicativo no Argo CD

- Na seção **Destination**, informar o seguinte:
  - Em **Cluster URL**, selecionar a URL
    - **https://kubernetes.default.svc**
  - Em **Namespace**, informar **default**
  - Para as demais opções manter o padrão
  - Após finalizar, clicar no botão **Create**
  - Clicar na opção **Sync**, do aplicativo criado
  - Clicar na opção **Synchronize**

CI/CD

Criando um repositório no Docker Hub

- Após logar, clicar no botão **Create repository**
- Em **Repository Name**, informar o nome desejado
- Manter a **Visibility** como **Public**
- Após finalizar, clicar no botão **Create**

CI/CD

Criando um repositório no GitHub

- Após logar, clicar no botão **+**
- No menu, clicar na opção **New repository**
- Em **Repository Name**, informar o nome desejado
- Manter a **Visibility** como **Public**
- Selecionar a opção **Add a README file**
- Após finalizar, clicar no botão **Create**

CI/CD

Estruturando o repositório no GitHub

- Será necessário criar um arquivo com o nome **src/index.html** e com o conteúdo:

```
<!DOCTYPE html>
<html>
<head>
<title>DevOps</title>
</head>
<body>

<h1>Hello DevOps World!</h1>
<p>Yeah! Another hello world...</p>

</body>
</html>
```

CI/CD

Estruturando o repositório no GitHub

- Será necessário criar um arquivo com o nome **Dockerfile** e com o conteúdo:

```
FROM nginx
COPY src/ /usr/share/nginx/html
```

CI/CD

Configurando as Secrets do GitHub Actions

- Selecionar o menu superior **Settings**
- Na seção **Security** expandir o menu **Secrets and variables**
- Selecionar o submenu **Actions**
- Adicionar três **Repository secrets**:
  - **DH_USER** -> usuário do Docker Hub
  - **DH_PASS** -> senha do Docker Hub
  - **DH_REPO** -> repositório do Docker Hub

CI/CD

Configurando o GitHub Actions

- Selecionar o menu superior **Actions**
- Escolher a opção **set up a workflow yourself**
- Copiar o código da URL **https://t.ly/rbYiD**
- Colar no arquivo aberto no GitHub
- Clicar no botão **Commit changes...**

CI/CD

Verificando a execução do GitHub Actions

- A partir de agora, todo **commit** realizado vai executar o **workflow** configurado
- Para verificar o resultado da execução do **workflow**, basta selecionar o menu superior **Actions**
- Se tudo estiver configurado corretamente, para cada commit será gerada uma imagem de docker e a imagem será subida para o Docker Hub

CI/CD

Criando mais um repositório no GitHub

- Criando um repositório para armazenar as configurações do kubernetes
- Após logar, clicar no botão **+**
- No menu, clicar na opção **New repository**
- Em **Repository Name**, informar o nome desejado
- Manter a **Visibility** como **Public**
- Selecionar a opção **Add a README file**
- Após finalizar, clicar no botão **Create**

CI/CD

Estruturando o novo repositório no GitHub

- Será necessário criar um arquivo com o nome **repo/service.yaml** e com o conteúdo:

```
apiVersion: v1
kind: Service
metadata:
  name: demo-ui
spec:
  type: NodePort
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: demo-ui
```

CI/CD

Estruturando o novo repositório no GitHub

- Será necessário criar um arquivo com o nome **deployment.yaml** e com o conteúdo:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: demo-ui
spec:
  replicas: 1
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: demo-ui
  template:
    metadata:
      labels:
        app: demo-ui
    spec:
      containers:
      - image: jucirodri/demo-ui:f4a0feb96f933c32840bfcdd3d4d4435f6cf04f1
        name: demo-ui
        ports:
        - containerPort: 80
```

CI/CD

Configurando as Secrets do GitHub Actions

- Voltar para o repositório do código-fonte
- Selecionar o menu superior **Settings**
- Na seção **Security** expandir o menu **Secrets and variables**
- Selecionar o submenu **Actions**
- Adicionar três **Repository secrets**:
  - **GH_TOKEN** -> token de acesso do GitHub
  - **GH_EMAIL** -> seu e-mail no GitHub
  - **GH_NAME** -> seu nome no GitHub

CI/CD

Gerando um token de acesso do GitHub

- Clicar na foto de perfil no canto superior direito e clicar no menu Settings
- Clicar no menu lateral Developer settings
- Expandir o menu Personal access tokens
- Clicar no submenu Tokens (classic)
- Clicar na opção Generate new token e no submenu Generate new token (classic)
- Informar um nome e selecionar repo, user e project e clicar no botão Generate token

CI/CD

Configurando o GitHub Actions

- Selecionar o menu superior **Actions**
- Selecionar o **workflow** no menu lateral
- Clicar no arquivo **main.yml**
- Selecionar a opção **Edit this file**
- Copiar o código da URL **https://t.ly/QhzzQ**
- Colar no arquivo aberto no GitHub
- Clicar no botão **Commit changes...**

CI/CD

Verificando a execução do GitHub Actions

- A partir de agora, todo **commit** realizado vai executar o **workflow** configurado
- Para verificar o resultado da execução do **workflow**, basta selecionar o menu superior **Actions**
- Se tudo estiver configurado corretamente, para cada commit, além de subir a imagem para o Docker Hub, será atualizado o arquivo de deployment do repositório de configuração com a nova imagem também

CI/CD

Criando mais um aplicativo no Argo CD

- Após logar, clicar no botão + New App
- Na seção General, informar o seguinte:
  - Em Application Name, informar demo-ui
  - Em Project Name, selecionar default
  - Manter a Sync Policy como Manual
  - Para as demais opções manter o padrão

CI/CD

Criando mais um aplicativo no Argo CD

- Na seção **Source**, informar o seguinte:
  - Em **Repository URL**, informar a URL
    - **https://github.com/jucirodri/k8s-apps.git**
  - Manter a **Revision** como **HEAD**
  - Em **Path**, informar **demo-ui**
  - Para as demais opções manter o padrão

CI/CD

Criando mais um aplicativo no Argo CD

- Na seção **Destination**, informar o seguinte:
  - Em **Cluster URL**, selecionar a URL
    - **https://kubernetes.default.svc**
  - Em **Namespace**, informar **default**
  - Para as demais opções manter o padrão
  - Após finalizar, clicar no botão **Create**
  - Clicar na opção **Sync**, do aplicativo criado
  - Clicar na opção **Synchronize**

FUTURO

FUTURO

DevOps e Inteligência Artificial

Existem vários tipos de IA usados em DevOps, incluindo:

- Aprendizado de máquina
- Processamento de linguagem natural
- Visão computacional
- Chatbots e assistentes virtuais.

REFERÊNCIAS

REFERÊNCIAS

Referências Bibliográficas

- Microsoft Azure. O que é o DevOps? DevOps explicado | Microsoft Azure. Disponível em: https://t.ly/kM9oj. Acesso em: 21 outubro 2023.

DEVOPS

ORQUESTRANDO O FUTURO DA ENTREGA DE SOFTWARE

jucimar.rodrigues@angellira.com.br
(49) 9 9840-4968

OBRIGADO!