Implantação e Atualização de Aplicações com Deployments

1. Clonagem de Recursos

Clonagem de recursos é uma técnica amplamente utilizada no gerenciamento de infraestrutura e aplicações, especialmente em ambientes de nuvem, como o Google Cloud. Essa prática permite duplicar a infraestrutura ou ambientes de aplicação para criar cópias idênticas de um recurso ou serviço.

- Quando Usar Clonagem de Recursos:
  - Ambientes de Teste: Criar uma réplica do ambiente de produção para testar novas funcionalidades sem afetar os usuários finais.
  - Desenvolvimento Paralelo: Desenvolvedores podem trabalhar em clones da infraestrutura sem impactar o ambiente principal.
  - Recuperação de Desastres: Clonar a infraestrutura para garantir que ela possa ser restaurada rapidamente em caso de falhas.

Passos para Clonar Recursos no Google Cloud:

1. Identificar os Recursos a Serem Clonados: Como máquinas virtuais, clusters Kubernetes, redes, ou buckets de armazenamento.

2. Usar Google Cloud Console ou CLI: Você pode clonar a configuração de VMs, discos, e outros recursos diretamente do Console ou usando o Google Cloud SDK.

Exemplo: Para clonar uma VM, você pode criar uma imagem da VM existente e, em seguida, criar uma nova VM baseada nessa imagem: `gcloud compute instances create clone-vm --image-family=clone-image --image-project=my-project`

Ao testar uma nova versão de uma aplicação crítica, você pode clonar o ambiente de produção para um ambiente de teste, garantindo que as atualizações possam ser verificadas sem impacto no serviço principal.

2. Configuração de Infraestrutura com Terraform

Terraform é uma ferramenta de código aberto para "Infraestrutura como Código" (IaC), permitindo que você defina, provisiona e gerencie infraestrutura de nuvem de maneira declarativa. No Google Cloud, o Terraform facilita a configuração e automação de recursos como máquinas virtuais, redes, e clusters Kubernetes.

Passos para Usar Terraform no Google Cloud:

1. Instalar o Terraform: Certifique-se de que o Terraform está instalado na sua máquina.

2. Criar um Arquivo de Configuração Terraform:

O arquivo `main.tf` define a infraestrutura que será criada. Exemplo para criar uma VM no Google Cloud:

```
provider "google" {
  project = "my-project"
  region  = "us-central1"
}
resource "google_compute_instance" "default" {
  name         = "terraform-instance"
  machine_type = "n1-standard-1"
  zone         = "us-central1-a"
  boot_disk {
    initialize_params {
      image = "debian-cloud/debian-10"
    }
  }
  network_interface {
    network = "default"
  }
}
```

3. Inicializar e Aplicar o Terraform:

Inicialize o Terraform para preparar o ambiente: terraform init

Aplique as configurações para criar os recursos: terraform apply

Se você precisa provisionar e gerenciar múltiplos clusters Kubernetes e VMs no Google Cloud, o Terraform permite que você defina toda essa infraestrutura em arquivos de configuração que podem ser versionados e reutilizados.

4.Geração de Registros

A geração de registros (logs) é um aspecto crítico para o monitoramento e depuração de aplicações e infraestruturas no Google Cloud. Os registros contêm informações detalhadas sobre as operações de cada componente da infraestrutura, permitindo que você monitore o desempenho e identifique problemas.

Tipos de Logs no Google Cloud:

- Google Cloud Logging: Coleta logs de todos os recursos do Google Cloud, como VMs, serviços gerenciados (GKE, Cloud Functions), e muito mais.

- Kubernetes Logs: GKE gera logs de cada contêiner, serviço e Pod em execução no cluster.

- Application Logs: Logs gerados por aplicações rodando dentro de contêineres ou VMs.

Passos para Geração de Registros:

1. Ativar o Logging para Recursos: O Google Cloud Logging é ativado por padrão para muitos serviços, mas pode ser configurado para recursos específicos como VMs ou clusters Kubernetes.

2. Uso de Logging na CLI:

Para visualizar logs de uma VM: `gcloud logging read "resource.type=gce_instance AND resource.labels.instance_id=INSTANCE_ID"`

Para acessar logs de Pods em um cluster Kubernetes: kubectl logs pod-name

Quando você tem uma aplicação em contêineres no GKE e encontra um problema de desempenho, os registros de logs detalhados podem ajudar a identificar se há falhas específicas em um Pod ou problemas de rede no cluster.

5. Visualização de Registros no Cloud Storage

Armazenar e visualizar logs de forma persistente é uma prática comum, especialmente em ambientes de produção. O Google Cloud permite

exportar e armazenar logs em buckets do Cloud Storage, onde eles podem ser analisados ou usados para auditoria.

Exportando Logs para o Cloud Storage:

1. Criação de um Bucket no Cloud Storage:

- No Google Cloud Console, crie um bucket que será usado para armazenar os logs.

2. Configuração da Exportação de Logs:

Configure uma exportação de logs para o bucket: `gcloud logging sinks create my-sink storage.googleapis.com/my-bucket --log-filter="severity>=ERROR"`

3. Verificação dos Logs no Cloud Storage:

- Acesse o bucket criado e veja os logs exportados, que serão armazenados como arquivos de texto ou JSON.

4. Visualização e Análise:

- Você pode abrir os arquivos de log diretamente no Cloud Storage ou baixá-los para análise local.

Suponha que você está monitorando uma aplicação com logs críticos de auditoria e precisa arquivar esses logs por um longo período para conformidade. Exportar esses logs para o Cloud Storage permite que você os armazene de forma segura e escalável.

Extra: Git, GitHub e Comandos Básicos do Git

1. O que é Git?

Git é um sistema de controle de versão distribuído amplamente utilizado que permite que várias pessoas trabalhem em um projeto simultaneamente, mantendo um histórico de todas as alterações feitas no código. Ele facilita o rastreamento de mudanças, a colaboração entre equipes e o gerenciamento de diferentes versões de um software.

- Características Principais do Git:
  - Distribuído: Cada desenvolvedor possui uma cópia completa do repositório.
  - Controle de Versão: Mantém o histórico de todas as mudanças feitas no código.
  - Branching e Merging: Facilita a criação de diferentes ramificações (branches) do projeto, permitindo o desenvolvimento paralelo e a fusão (merge) dessas mudanças.

Um time de desenvolvedores trabalhando em um aplicativo pode usar Git para garantir que todos possam colaborar simultaneamente em diferentes funcionalidades, integrando as mudanças sem conflitos.

2. O que é GitHub?

GitHub é uma plataforma de hospedagem de código baseada em Git, que permite a colaboração e o compartilhamento de projetos de software. Ele fornece uma interface web amigável, além de ferramentas para controle de versão, automação, e integração contínua.

- Funcionalidades do GitHub:
  - Repositórios Públicos e Privados: Os projetos podem ser compartilhados abertamente ou mantidos privados.

- Forks e Pull Requests: Facilitam a colaboração em projetos de código aberto, permitindo que usuários façam cópias (forks) e proponham mudanças (pull requests).

- GitHub Actions: Permite automação de pipelines CI/CD diretamente no repositório.

Uma empresa que desenvolve software pode usar GitHub para hospedar seus projetos de código privado, enquanto projetos de código aberto, como bibliotecas ou ferramentas, podem ser compartilhados publicamente com a comunidade de desenvolvedores.

3. Comandos Básicos do Git

Aqui estão os comandos mais usados no Git para gerenciar repositórios e colaborar em projetos:

Inicialização e Configuração

1. git init:

Inicializa um novo repositório Git em um diretório.

```
git init
```

2. git clone [URL]:

Faz uma cópia local de um repositório Git remoto.

```
git clone https://github.com/usuario/projeto.git
```

3. git config:

Configurar opções do Git, como o nome e email do usuário.

```
git config --global user.name "Seu Nome"
```

```
git config --global user.email "seu.email@exemplo.com"
```

Trabalhando com Alterações

4. git status:

Mostra o status atual do repositório, exibindo arquivos modificados e que não estão em stage (prontos para commit).

```
git status
```

5. git add [arquivo]:

Adicionar arquivos ou mudanças ao stage para o próximo commit.

```
git add arquivo.txt
```

6. git commit -m "[mensagem]":

Criar um commit com as mudanças em stage e uma mensagem descritiva.

```
git commit -m "Adiciona nova funcionalidade X"
```

Branching e Merging

7. git branch:

Exibe todas as branches no repositório e indica a branch atual.

```
git branch
```

8. git checkout [branch]:

Troca para uma branch existente ou cria uma nova branch.

```
git checkout -b nova-branch
```

9. git merge [branch]:

Faz a fusão de uma branch específica com a branch atual.

```
git merge feature-branch
```

Trabalhando com Repositórios Remotos

10. git remote add origin [URL]:Conecta um repositório local a um repositório remoto.

```
git remote add origin https://github.com/usuario/projeto.git
```

11. git push [remote] [branch]:

Envia os commits da branch local para o repositório remoto.

```
git push origin main
```

12. git pull [remote] [branch]:

Baixa as últimas mudanças do repositório remoto e as integra à branch atual.

```
git pull origin main
```

Revisão de Histórico e Reversão

13. git log:

Exibe o histórico de commits no repositório.

```
git log
```

14. git revert [commit]:

Cria um novo commit que reverte as mudanças de um commit anterior.

```
git revert abcd123
```

15. git reset [commit]:

Desfaz commits locais, mantendo ou removendo as mudanças.

```
git reset --hard abcd123
```

4. Integração entre Git e GitHub

O fluxo de trabalho típico ao usar Git e GitHub inclui os seguintes passos:

1. Clone o Repositório: Use git clone para baixar o código do GitHub para sua máquina.

2. Crie e Mude para uma Nova Branch: Trabalhe em uma nova funcionalidade ou correção de bug em uma branch separada.

3. Adicione e Commite as Mudanças: Depois de concluir a tarefa, use git add e git commit para salvar as mudanças.

4. Envie as Mudanças para o GitHub: Use git push para enviar as alterações para o repositório remoto no GitHub.

5. Pull Request: Caso esteja colaborando em um projeto de código aberto, faça um Pull Request no GitHub para sugerir suas mudanças ao mantenedor do projeto.

Conteúdo Bônus

Título: Deployments
Plataforma: Kubernetes.io
Descrição: Explora o uso de deployments no Kubernetes para gerenciar o ciclo de vida de aplicações, incluindo atualizações e rollbacks.

Referências Bibliográficas

BASSO, D. E. **Administração de Redes de Computadores.** Contentus, 2020.

KUROSE, J. F.; ROSS, K. W. **Redes de Computadores e a Internet: Uma Abordagem Top-Down**. 8. ed. Pearson, 2021.

MARINHO, A. L.; CRUZ, J. L. da. (Orgs.). **Desenvolvimento de Aplicações para Internet**. 2. ed. Pearson, 2020.

PUGA, S.; RISSETTI, G. **Lógica de Programação e Estruturas de Dados, com Aplicações em Java**. 3. ed. Pearson, 2016.

ROHLING, L. J. **Segurança de Redes de Computadores**. Contentus, 2020.

SILVA, C. F. da. **Projeto Estruturado e Gerência de Redes**. Contentus, 2020.

TANENBAUM, A. S.; FEAMSTER, N.; WETHERALL, D. J. **Redes de Computadores**. 6. ed. Pearson, 2021.

TOCCI, R. J.; WIDMER, N. S.; MOSS, G. L. **Sistemas Digitais: Princípios e Aplicações**. 12. ed. Pearson, 2018.

Ir para exercício