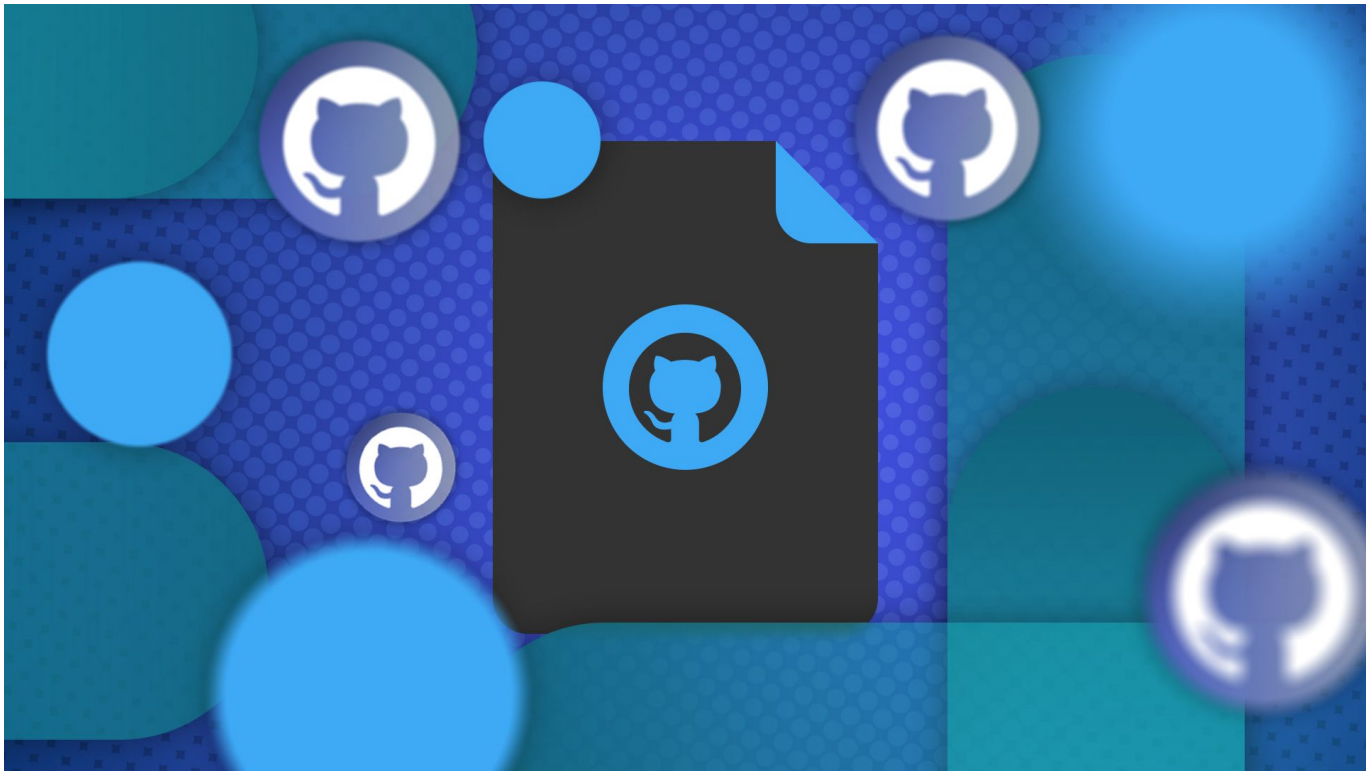

O que você vai ler:

Resumo

- Gerenciar dotfiles com git pode economizar tempo e fornecer uma opção de backup robusta.

- Armazenar dotfiles em um sistema de controle de versão (VCS) como o git pode garantir uma configuração consistente em várias máquinas.

- Usar o GitHub para hospedar dotfiles permite fácil compartilhamento e colaboração.

Dotfiles são uma forma acessível e poderosa de configurar seu sistema Linux. Mas como você controla todos eles e os reutiliza quando necessário? Tente isso.

O que são arquivos de pontos?

No Linux, qualquer arquivo cujo nome comece com “.” é um arquivo oculto. Por padrão, ele não será exibido no gerenciador de arquivos ou em uma listagem de arquivos na linha de comando.

Alguns programas Linux usam arquivos ocultos para configuração, muitas vezes despejando-os em seu diretório inicial. Esta é uma configuração útil porque mantém a configuração fora do caminho e ao mesmo tempo garante que ela ainda esteja acessível. Como esta configuração está em arquivos de texto simples, é fácil de ler e editar. Você também pode usar o conjunto de ferramentas de linha de comando do Linux para trabalhar com a configuração do seu sistema.

Exemplos comuns de dotfiles incluem:

- .bashrc, .zshrc
- .exrc
- .gitconfig
- .npmrc

Como o Git ou o GitHub podem ajudar?

Dotfiles são ótimos, mas são específicos do sistema. Quando precisar substituir seu computador, usar um dispositivo secundário ou acessar um servidor remoto, você poderá

configurar tudo novamente.

Armazenar seus dotfiles em um VCS (Sistema de Controle de Versão) pode ajudá-lo a evitar essa tarefa repetitiva, permitindo reutilizar instantaneamente sua configuração em outra máquina. Basta verificar seu repositório e você obterá os mesmos aliases de shell, temas familiares e comportamento consistente.

Além do mais, armazenar dotfiles no git é uma opção robusta de backup. Você pode até inspecionar o histórico do seu repositório para descobrir quando — e por que — você alterou uma configuração específica. Em um ambiente colaborativo, você pode até compartilhar seus dotfiles via git para garantir que todos os membros da equipe tenham um ambiente consistente.

Para esse propósito, o GitHub é a nata da cultura. Se você tiver outro local para hospedar seu repositório git, certamente poderá fazer isso, mas o GitHub torna isso muito mais fácil.

A melhor maneira de gerenciar seus Dotfiles com Git e GitHub

Primeiro, entenda que qualquer forma de armazenar seus dotfiles no git será uma grande vitória. Existem detalhes específicos sobre exatamente a melhor maneira de fazer isso, mas se você puder armazenar um arquivo no git, atualizá-lo e verificá-lo, você se beneficiará significativamente ao gerenciar seus dotfiles dessa maneira.

No entanto, a abordagem a seguir é amplamente recomendada online e funciona para mim. Esta configuração específica deve ajudá-lo a manter tudo sincronizado com o mínimo esforço.

Configure um repositório vazio e alguns andaimes

Como seu diretório inicial provavelmente contém muitas coisas que você não deseja em seu repositório dotfiles, é melhor evitar uma configuração padrão. Em vez disso, você pode gerenciar seus dotfiles em um repositório vazio.

Um repositório vazio é como um repositório normal sem os arquivos reais do projeto. Ele possui todos os metadados do git que descrevem o histórico desses arquivos, mas não possui os próprios arquivos. Os arquivos podem estar em outro lugar – em seu diretório de trabalho – você apenas usará o repositório vazio para controlá-los.

Comece criando um repositório vazio em um novo local, por exemplo:

```
mkdir $HOME/.dotfiles
git init --bare $HOME/.dotfiles
```

Ao trabalhar com este repositório, você precisará fornecer um diretório de trabalho (para os arquivos) e um diretório git (para o próprio repositório):

```
git --work-tree=$HOME --git-dir=$HOME/.dotfiles ...
```

Em vez de digitar isso toda vez que usar o git, faz sentido configurar um alias. Você também pode fornecer o caminho para o próprio repositório vazio para poder usá-lo em qualquer diretório:

```
alias dotfiles="/usr/bin/git --git-dir=$HOME/.dotfiles --work-
tree=$HOME"
```

Armazene seus arquivos de ponto

Comece identificando um dotfile que você deseja controlar a versão.

Você pode então executar estes comandos para iniciar o controle de versão do seu arquivo .bashrc, por exemplo:

```
cd $HOME
dotfiles add .bashrc
dotfiles commit -m "Bash run control file"
```

Além de usar o alias dotfiles em vez do comando git simples, você pode usar o git para rastrear esses arquivos como faria normalmente. Na verdade, essa abordagem é um pouco mais fácil porque você pode executar um comando como “dotfiles log” de qualquer diretório.

Carregue seu repositório no GitHub

Você pode achar conveniente hospedar seu repositório em um provedor como o GitHub. Isso pode facilitar o compartilhamento do acesso aos seus dotfiles, especialmente de máquinas em uma rede diferente. É fácil fazer isso, mesmo com um repositório existente:

1. Comece na página Criar um novo repositório.
2. Insira um nome de repositório.
3. Escolha um repositório Público ou Privado; Privado é provavelmente o melhor (veja abaixo).
4. Clique em Criar repositório.

Neste ponto, será exibida uma tela com instruções de configuração. Para enviar seu repositório existente, basta executar estes dois comandos:

```
dotfiles remote add origin https://github.com//.git
dotfiles push -u origin main
```

Onde está o seu nome de usuário do GitHub e o nome que você escolheu para o seu repositório.

Tenha muito cuidado ao enviar seu repositório para o GitHub: seus dotfiles podem conter dados confidenciais. Idealmente, você deve evitar enviar arquivos que contenham senhas para qualquer repositório. Se você não puder evitar, considere pelo menos usar um repositório GitHub privado; você precisará pagar por isso, no entanto.

Use em outro sistema

Para compartilhar seus dotfiles em outra máquina, você precisará repetir os processos acima e verificar o repositório vazio. Em particular, isso significa que há duas etapas importantes. Primeiro, verifique uma cópia simples do seu repositório:

```
cd $HOME
```

```
git clone --bare https://github.com//.git
```

Normalmente, isso fará check-out em um diretório chamado .git. Uma vez verificado, você está livre para renomeá-lo.

Recrie o alias do wrapper git que você está usando:

```
alias dotfiles="/usr/bin/git --git-dir=$HOME/.dotfiles --work-tree=$HOME"
```

Agora você pode preencher seu diretório de trabalho – seu HOME – com seus dotfiles controlados por versão:

```
dotfiles checkout
```

Neste ponto, você poderá ver um erro sobre a substituição dos arquivos da árvore de trabalho. Isso ocorre porque você provavelmente já possui dotfiles antigos ou padrão como .bashrc. Basta removê-los ou movê-los e repetir a verificação.

O controle de versão de seus dotfiles evitará muitos problemas ao atualizar ou trocar de sistema. Você também poderá verificar um histórico completo e ver quando mudou o quê e por quê.