Linux (Ubuntu)

1. Introdução

Neste tutorial, vamos aprender a configurar o ambiente de desenvolvimento do nosso projeto no Ubuntu, abordando a instalação e configuração de:

- Docker
- NVM (Node Version Manager)
- VS Code
- Git e GitHub

O tutorial também pode ser seguido em outras distribuições baseadas em Ubuntu, que podem ser conferidas neste link. Recomenda-se a versão 22.04 LTS ou superior.

> 🔴 **Atenção! Essa é uma aula com conteúdo de nível avançado. Caso você seja iniciante, recomendamos que siga com o desenvolvimento do projeto no ambiente do Codespaces. Mais pra frente, quando estiver mais confortável, você pode voltar aqui e configurar o seu ambiente local, sem prejuízo algum.**

2. Docker no Linux

O Docker é um recurso fantástico e nós teremos a oportunidade de nos aprofundarmos mais nele ao chegarmos no `Dia 17`, quando formos configurar o nosso banco de dados local. Mas, por enquanto, basta dizer que ele nos permite executar programas em ambientes isolados e padronizados, conhecidos como contêineres, com a garantia de que esses programas funcionarão da exata mesma maneira em qualquer máquina, independentemente das configuração de hardware ou software desta.

No Linux, a instalação e o uso do Docker são mais simples do que no Windows, porque não é necessário criar nenhuma camada adicional de virtualização para que ele funcione. Isso acontece porque a "casa" do Docker é justamente o Linux — foi nesse ambiente que ele foi projetado originalmente e é nele que estão disponíveis, de forma nativa, todos os recursos dos quais depende. Isso simplifica bastante a configuração do ambiente e reduz o consumo de recursos da máquina.

2.1 O backend do Docker no Linux

No Windows, o Docker precisa de uma camada extra (Hyper-V ou WSL) para simular um ambiente Linux. No Linux, essa camada já existe: o próprio kernel do sistema oferece os recursos de isolamento e gerenciamento de processos que o Docker utiliza, como namespaces e cgroups. Em outras palavras, enquanto no Windows o Docker depende de uma máquina virtual para rodar, no Linux ele roda "direto na máquina".

2.2 Docker Engine e Docker Desktop

O Docker Engine é o núcleo da ferramenta — é ele quem cria, executa e gerencia os contêineres. Já o Docker Desktop é uma solução mais completa, que inclui o Docker Engine e adiciona uma interface gráfica para facilitar algumas configurações.

2.3 Instalando o Docker Desktop no Ubuntu

O Docker Desktop no Linux roda dentro de uma **máquina virtual (VM)**, e para que essa VM funcione de forma eficiente, é necessário que o sistema tenha suporte à **virtualização KVM**.

O **KVM (Kernel-based Virtual Machine)** é uma tecnologia de virtualização do Linux que permite transformar o kernel do sistema em um hipervisor. Em outras palavras, ele possibilita que você execute máquinas virtuais diretamente no Linux, aproveitando os recursos do processador, como múltiplos núcleos e instruções de virtualização, sem precisar de software adicional pesado.

Para que o Docker Desktop funcione corretamente, o KVM precisa estar ativo.

Na maioria das distribuições modernas, se o processador suporta virtualização, o módulo KVM já é carregado automaticamente. Muito provavelmente, você não precisará fazer nenhuma configuração adicional nesse ponto e pode seguir direto para a instalação do Docker Desktop. No entanto, se for necessário, você pode carregar e configurar o KVM manualmente seguindo os passos deste link.

Atenção! É importante destacar que o Docker exige, no mínimo, **4 GB de RAM** para funcionar corretamente.

Primeiro, precisamos adicionar a chave GPG do Docker, que permite ao sistema verificar a autenticidade dos pacotes do Docker:

```
sudo apt update
sudo apt install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

Explicando cada comando:

- `sudo apt update` — atualiza a lista de pacotes disponíveis.
- `sudo apt install ca-certificates curl` — instala certificados SSL e o `curl`, necessários para baixar arquivos via HTTPS.
- `sudo install -m 0755 -d /etc/apt/keyrings` — cria o diretório `/etc/apt/keyrings` com permissões adequadas para armazenar chaves.
- `sudo curl -fsSL <URL> -o /etc/apt/keyrings/docker.asc` — baixa a chave GPG oficial do Docker e salva no diretório criado.
- `sudo chmod a+r /etc/apt/keyrings/docker.asc` — garante que todos os usuários possam ler a chave.

Depois de adicionar a chave, precisamos configurar o repositório oficial do Docker no sistema:

```
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https:
  $(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

E executar novamente o comando:

```
sudo apt update
```

Explicando:

- `dpkg --print-architecture` — identifica a arquitetura do seu processador (amd64, arm64, etc).
- `$(. /etc/os-release && echo "${UBUNTU_CODENAME:-$VERSION_CODENAME}")` — detecta a versão do Ubuntu que você está usando (como `jammy` para 22.04).
- O comando `tee` adiciona o repositório ao sistema de forma segura.

Antes de continuar, é uma boa prática garantir que todos os pacotes estejam atualizados:

```
sudo apt upgrade -y
```

Agora, baixe o arquivo `.deb` do Docker Desktop pelo site oficial. Depois, abra o terminal, navegue até a pasta onde o arquivo foi salvo e execute:

```
sudo apt update
sudo apt install ./docker-desktop-amd64.deb
```

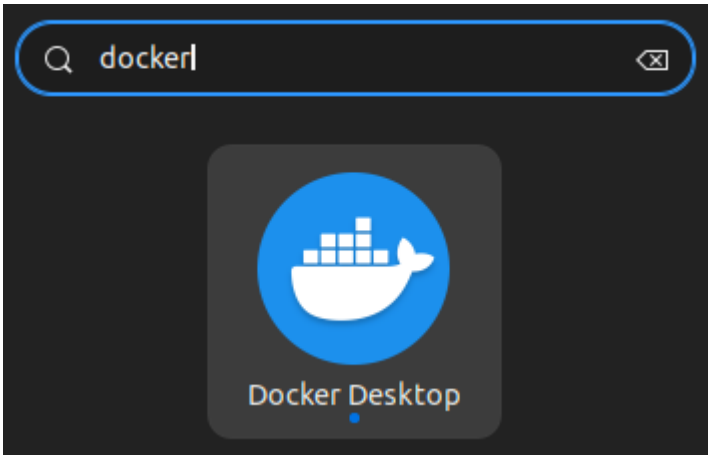
- O `apt install ./arquivo.deb` instala pacotes `.deb` manualmente.

- O pacote `.deb` inclui scripts de pós-instalação que configuram automaticamente o Docker Desktop.

Durante a instalação, pode aparecer um aviso de permissão como esse abaixo. Você pode **ignorar essa mensagem**.

```
N: Download is performed unsandboxed as root, as file '/home/user/Downloads/docker-t
```

Depois de instalado, você pode iniciar o Docker Desktop pelo seu ambiente gráfico. Abra o menu de aplicativos e localize **Docker Desktop**.



De maneira alternativa, você pode executar o seguinte comando no terminal:

```
systemctl --user start docker-desktop
```

🔴 Atenção: se estiver com problemas para inicializar o Docker após a instalação, dá uma olhada nesse comentário do aluno `MardoqueuLS` .

3. NVM no Ubuntu

A instalação do NVM é bem mais simples do que a do Docker. De acordo com a documentação oficial, a forma mais prática é executar o seguinte comando:

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.40.3/install.sh | bash
```

Em seguida, é necessário fechar o terminal e abri-lo novamente para que o NVM esteja disponível.

Para verificar se a instalação foi bem-sucedida, execute:

```
nvm --version
```

Na Pista Lenta **A Fundação**, o Filipe irá orientar a instalação do Node.js utilizando o NVM.

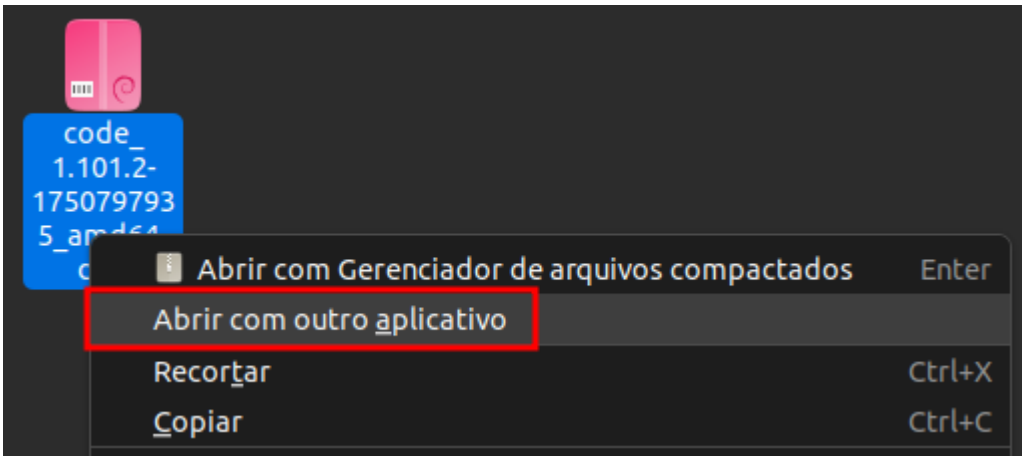
4. Instalando o editor de código

Como editor de código, nós recomendamos a utilização do VS Code. Mas caso você tenha algum outro que seja da sua preferência, sinta-se à vontade para usá-lo.

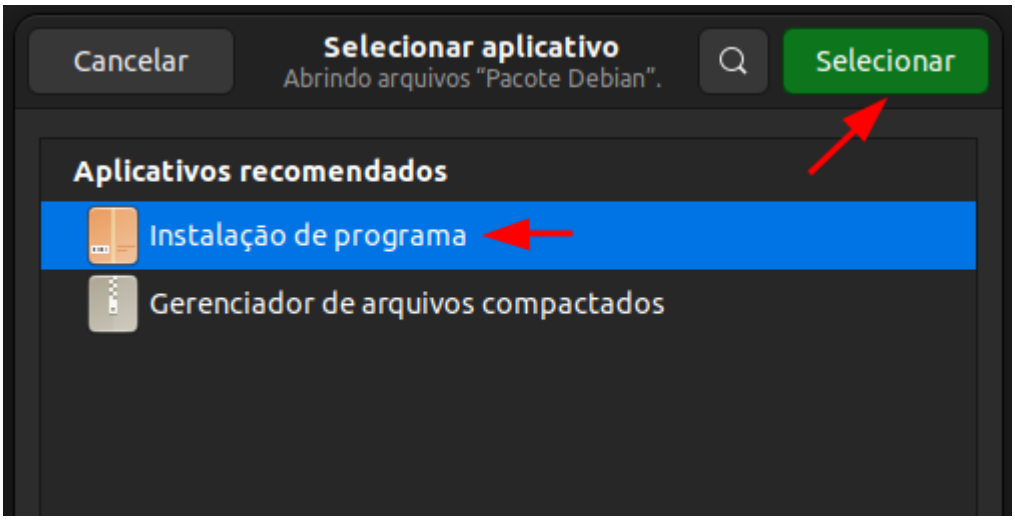
Para baixar o VS Code, acesse o site oficial e clique em **Download for Linux (.deb)**.

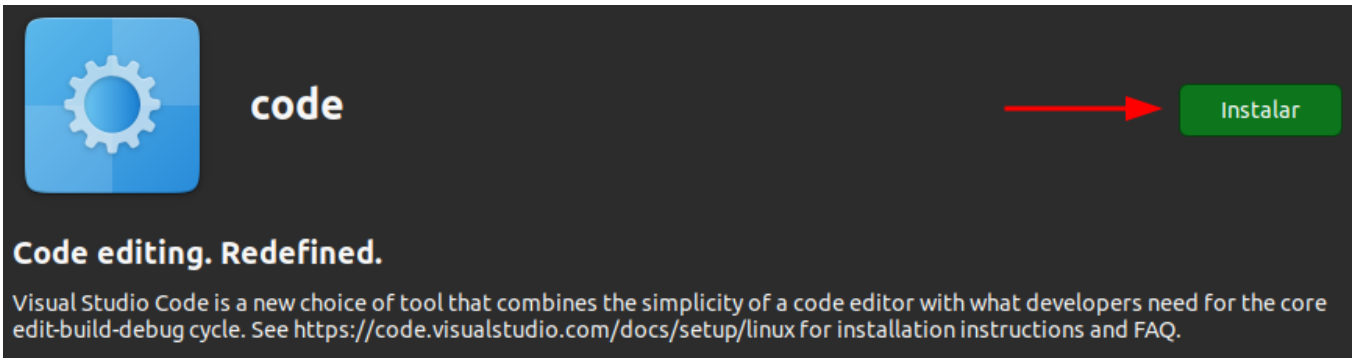
4.1 Instalando via interface gráfica

Localize o arquivo `.deb` baixado, clique com o botão direito sobre ele e selecione **Abrir com outro aplicativo**.

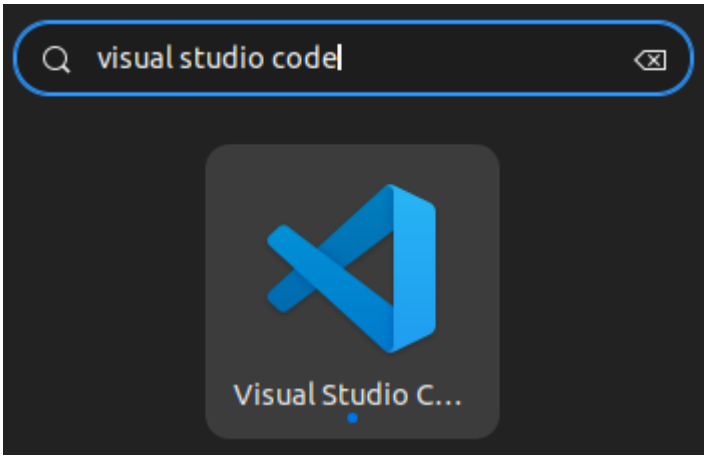


Escolha a opção **Instalação de Programas** e, na janela que se abrir, clique em **Instalar**. Pode ser necessário informar sua senha de usuário. Aguarde até que a instalação seja concluída.





Após a instalação, o VS Code estará disponível na seção de aplicativos.



4.2 Instalando via linha de comando

No terminal, localize o arquivo `.deb` baixado e execute o seguinte comando:

```
sudo apt update
sudo apt install ./nome-do-arquivo.deb
```

Caso a instalação retorne erro de dependência, execute o comando:

```
sudo apt install -f
```

E pronto, o VS Code está instalado com sucesso!

5. Configurando o Git e GitHub

No Ubuntu, o Git vem instalado por padrão. Você pode conferir isso executando o comando:

```
git --version
```

Caso não esteja instalado, basta executar:

```
sudo apt install git
```

Para configurar o Git e o GitHub no Ubuntu para o nosso projeto, você pode seguir o passo a passo da seção `5. Configurando o Git e GitHub` da aula do ambiente Windows, disponibilizada pelo `Andrei` .

Referências

- https://docs.docker.com/desktop/setup/install/linux/
- https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository
- https://docs.docker.com/desktop/setup/install/linux/ubuntu/
- https://github.com/nvm-sh/nvm

Concluir aula 15xp Próxima aula >

Responder

`DiegoSouzaReis` 22 dias atrás

1

Pessoal, troquei do Windows para o Zorin OS recentemente e queria fazer algo diferente do que fazia no Windows. Uma das coisas é conseguir tentar replicar o ambiente de produção de forma local. Ex:
Em produção utiliza:

1. 2 CPUs
2. 4 GB de RAM
   Quero conseguir replicar isso localmente para conseguir otimizar melhor meu código. Só que quero fazer isso por projeto.
   Quero conseguir simular também o quanto de recursos tenho disponível no meu banco de dados.
   Se alguém já faz isso ou tiver algum artigo que conheça que fale sobre isso, me informe pf. Quero ver alguns vídeos também sobre isso.

Responder

`LLuis` 29 dias atrás

4

Pra quem quer uma alternativa ao DockerDesktop tem o Podman, você instala o Docker Engine depois o Docker Compose e consegue gerenciar os containers pela interface do Podman

Responder

LuanGregati equipe 28 dias atrás

1 Ótima dica, LLuis !

Responder

Rennan 1 mês atrás

2 Para quem teve o erro "docker-desktop : Depende: docker-ce-cli mas não é instalável" durante a instalação do docker desktop, pode seguir este passo a passo da documentação do docker (link) para adicionar os repositórios no apt. Após isso, o apt vai instalar as dependências que estão faltando automaticamente quando executar o pacote .deb.

Responder

LuanGregati equipe 1 mês atrás

2 Valeu pela dica, Rennan !

Responder

Gutem 1 mês atrás

2 O denzylegacy citou o Arch (btw, I used to use Arch ;)), atualmente tem o Omarchy, distro Linux baseada no Arch, criado pelo DHH, criador do Rails em "Ruby on Rails": https://omarchy.org/

Responder

DiasGabriel 1 mês atrás

1 Também estou utilizando essa distro, porém a apenas 4 dias. Tem algumas dicas ou pacotes recomendados?

Responder

denzylegacy 1 mês atrás

3 Para quem também usa o Arch ou uma distro derivada, recomendo este artigo sobre como instalar os requisitos necessários para o desenvolvimento do projeto:

https://www.atlantic.net/dedicated-server-hosting/how-to-install-and-use-node-js-on-arch-linux/

O autor usa o `pacman`, mas você pode optar pelo `yay` ou `paru` pois boas alternativas. Particularmente, tenho preferência pelo `yay`.

Responder

`DiasGabriel` 1 mês atrás

2 Também estou utilizando essa distro, porém a apenas 4 dias. Tem algumas dicas ou pacotes recomendados?

Responder

`denzylegacy` 1 mês atrás

1 DiasGabriel, você instalou o Arch manualmente ou utilizou um script como o archinstall? Qual DE você escolheu?

Quanto aos pacotes, isso realmente depende do que você pretende fazer. Uma dica importante é ter cautela ao executar atualizações completas do sistema (tipo `pacman -Syu`), já que o Arch (e suas variantes) é uma distribuição rolling release. Portanto os pacotes não passam por testes extensivos antes de serem disponibilizados (e isso pode causar instabilidades).

Além disso, eu recomendo que você explore a flexibilidade da distro com um ricing (eu não sou desses, mas vale a aventura). No mais... dedique um tempo para dominar o terminal, pois isso poderá te ensinar muito.

Responder

`LuanGregati` equipe 1 mês atrás

2 Muito obrigado por compartilhar, `denzylegacy`!

Responder

`Prize` 2 meses atrás

2 adorei que temos um tutorial para nós do time do pinguim! as vezes faz falta em alguns cursos!

Responder

LuanGregati equipe 2 meses atrás

2 Shooow, que bom que curtiu 😍🎉

Responder

msilva 2 meses atrás

3 Seria bacana termos também uma menção ou nota de como contornar o erro `Your CPU does not support KVM extensions` no meu caso eu precisei ativar o `SVM Mode` na bios.

Responder

LuanGregati equipe 2 meses atrás

1 Boa, `msilva` ! Vou ver isso 🫡

Muito obrigado pelo feedback.

Responder

JuniorC07 2 meses atrás

2 São ferramentas que uso no dia-adia, então já tinha tudo configurado, a unica diferença é que uso o n invés do nvm

Responder

LuanGregati equipe 2 meses atrás

1 Show, `JuniorC07` 💪

Responder

rivaroles 2 meses atrás

3 Estou usando Arch e foi uma batalha...

Responder

viniciushenning 2 meses atrás

1 Força meu bom! Eu acabei abandonando o Arch devido a algumas dificuldades de integração com a placa de vídeo hahahah

Responder

Lucass2 2 meses atrás

1 eu abandonei o fedora pela mesma situação

Responder

maykcaldas 2 meses atrás

1 Brabo hahaha Tá há quanto tempo com o arch?

Responder

rivaroles 2 meses atrás

1 Tô usando faz mais ou menos 6~8 meses haushaush

Responder

snagh1 2 meses atrás

1 Eu ri com muito respeito. Eu fiquei imaginando que se eu, que estou usando o Mint, quase não dou conta, imagina outros sistemas mais complexos.

Responder

Alistair 2 meses atrás

1 Excelente OS está bem, eu usei o Mint por alguns anos e posso garantir que ele me deu a bsae necessária para que hj eu pudesse ingressar em outras distro Linux. portanto, aproveite a viagem no Mint.

Responder

LuanGregati equipe 2 meses atrás

1 Conseguiu, `rivaroles` ?

Responder

rivaroles 2 meses atrás

2 Consegui sim, só fiquei meio perdido, mas acho que deu tudo certo por aqui

Responder

LuanGregati equipe 2 meses atrás

1 Show! Qualquer coisa conte com a gente 🤜

Responder

DevNick 2 meses atrás

3 Estou usando a distro linux Aurora-DX, ela já vem com o docker e o podman instalados. Não recomendo para novatos, as coisas vem instalados para você, mas várias coisas são diferentes do tutorial do Champs. Eu vou continuar apanhando nele, mas aprendendo também.

Responder

LuanGregati equipe 2 meses atrás

1 Show, DevNick ! O que vale é o aprendizado 💪

Responder

MardoqueuLS 3 meses atrás

8 **Tudo que está aqui agora está configurado S2**

Vou deixar um comentátio que talvez ajude alguém no futuro.
Ao abrir meu Docker Desktop tive uma mensagem de erro de **KVM**. Praticamente nada que eu fazia dava certo pra resolver, até eu me dar conta de que uma das mensagens que recebi de erro era sobre falta de suporte do processador. Lembrei que essa opção de Virtualização tem que estar habilitada na BIOS (caso seu hardware tenha o suporte).

Fui na BIOS e "bingo", a virtualização estava desabilitada. Após habilitar voltei ao sistema, iniciei o Docker Desktop e a mensagem de falta de suporte a Virtualização não apareceu mais e o Docker está funcionando normalmente.

Responder

LuanGregati equipe 3 meses atrás

1 Muito obrigado por compartilhar isso, MardoqueuLS 💪

Responder