Como usar o servidor da Fábrica de Software

Minicurso

Introdução

- O porquê deste minicurso;
- Objetivos
  - Orientar em como usar o servidor para colocar um projeto qualquer em operação.
- O que será aprendido
- Pauta
  - O que é Linux
  - Como conectar-se ao servidor via SSH
  - Como usar o gerenciador de pacotes do Ubuntu
  - Como colocar um projeto em operação

Linux

- O que é Linux?
- Linus Torvalds
- *Open source*
- Distribuições Linux
    - Debian - Ubuntu - Debian/Ubuntu *based distros*
    - Arch Linux
    - RHEL - Fedora

Linux - Curiosidades

- Todos os top 500 supercomputadores usam Linux
- 85% de todos os *smartphones* usam Linux (Android)
- 95% dos servidores dentre os top 1 milhão usam Linux
    - Facebook, Twitter, Youtube, etc

SSH - Secure Shell

- Protocolo para acesso remoto seguro
- Cliente-servidor
- Permite a execução de comandos de maneira remota via *shell*
- Porta padrão: 22
- Comunicação entre cliente e servidor criptografada
- *Public Key Authentication*

SSH - Como usar

- $ ssh *username@domain*
- Explicitando a porta:
  - $ ssh -p *port username@domain*
- Cada projeto terá sua porta de acesso e nome de usuário

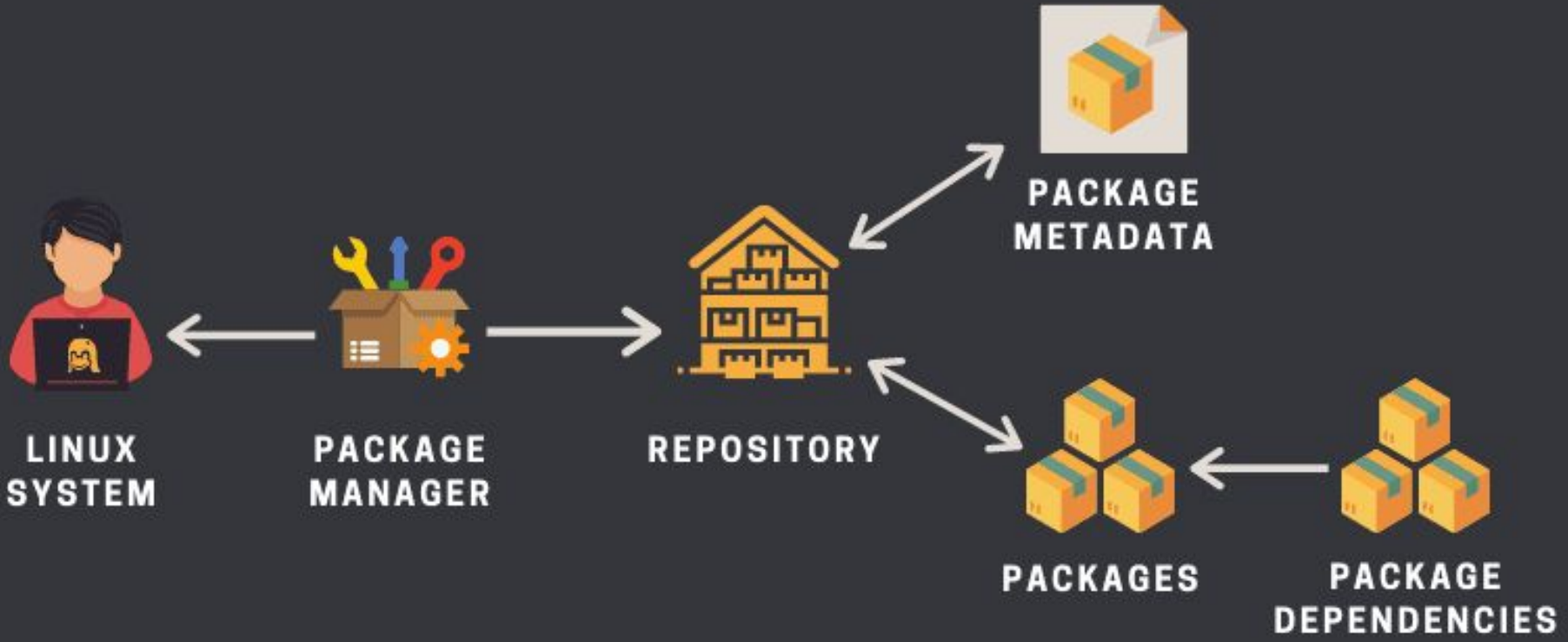
Package Manager

- Ferramenta que permite instalar, remover, atualizar configurar e gerenciar pacotes (programas)
- Pode ser usado via CLI (*Command Line Interface*) ou via interface gráfica (porém, o servidor não possui nada gráfico)
- Não é exclusivo do Linux. Exemplos: pip, npm, yarn
- Package manager do Ubuntu: apt (apt-get)

Package manager - Como funciona

- Repositórios que contêm os pacotes e seus metadados
- Cache local
- Dependências
- *Sources list*

IT'S FOSS


PACKAGE METADATA
LINUX SYSTEM
PACKAGE MANAGER
REPOSITORY
PACKAGES
PACKAGE DEPENDENCIES

Package Manager - Como funciona

- $ sudo apt update (atualiza cache local)
- $ sudo apt install *package_name* (instala o pacote e suas dependências)
- $ sudo apt upgrade (atualiza os pacotes instalados)
- $ sudo apt autoremove (remove pacotes não utilizados)
- Além de instalar, remover e atualizar pacotes, o *package manager* pode fazer mais coisas
- É possível travar um programa em uma versão específica, impedindo que seja atualizado a outra versão
    - $ sudo apt-mark hold *package_name*

Usando o package manager

- Atualizar cache local
  - $ sudo apt update
- Instalar o editor de texto micro e o vim e instalar o monitor htop
  - $ sudo apt install micro
  - $ sudo apt install vim
  - $ sudo apt install htop
  - Ou, $ sudo apt install micro vim htop
- Remover algum desses pacotes
  - $ sudo apt remove vim

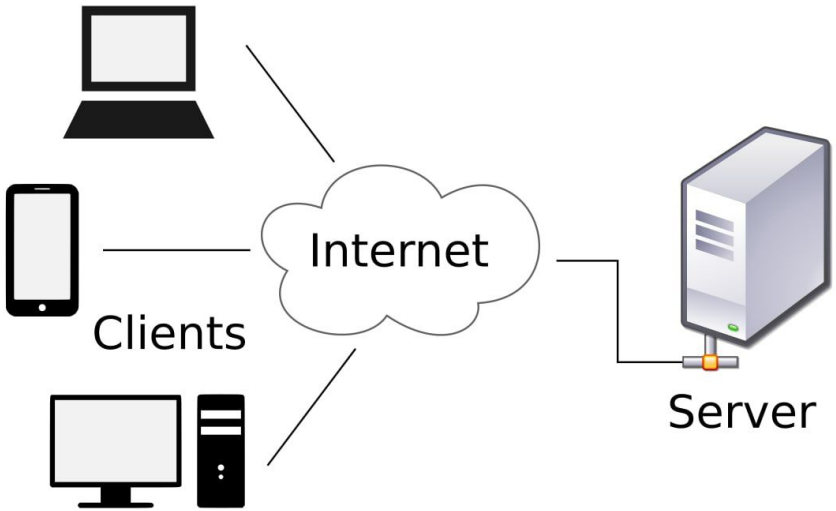
Visão geral sobre a operação de um projeto

- Vamos criar um novo projeto em Laravel
- O projeto conterá apenas uma simples conexão com o banco de dados MySQL
- Servirá como norte para qualquer outro projeto

Cliente-Servidor

- Modelo de aplicações distribuídas
- Define dois papéis: o de cliente e o de servidor
- Servidor: oferece um serviço
- Cliente: consome um serviço
- A comunicação é sempre iniciada por um cliente
- O servidor deve estar sempre esperando pedidos de algum cliente (LISTENING)
- Exemplo de aplicações: e-mail, WWW

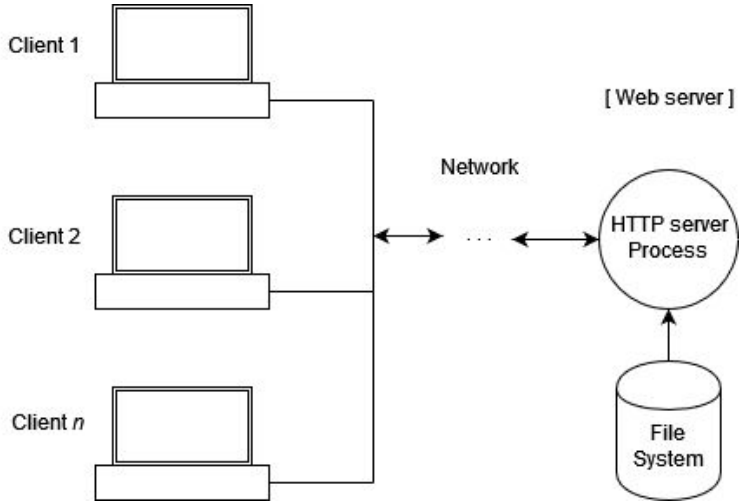
Cliente-Servidor



Client Server Network

Web Server

- Um programa que aceita requisições HTTP/HTTPS
- Fica sempre esperando que um cliente inicie uma comunicação
- Exemplo de web servers: Nginx, Apache



Banco de dados

- De modo geral, uma coleção organizada de dados
- Pode ser visto como um serviço
- Usa o modelo cliente-servidor
- O cliente será o próprio sistema

Dependências

- Todo software é composto de demais “peças” que são outros softwares
- Desse modo, um software depende de diversos componentes
- Sem a instalação das dependências, um software não pode operar
- Um passo fundamental é instalar todas as dependências
- O *package manager* facilita a instalação e gerenciamento de dependências

Firewall

- Determina o que entra, o que passa e o que sai
- Monitora e controla o tráfego de rede
- Determina por onde os pacotes podem entrar
- Para que o servidor possa receber requisições via HTTP, a porta 80 deve estar liberada para a entrada de pacotes
- Recomendação: UFW (Uncomplicated Firewall), um programa para criar e gerenciar regras no *firewall* de maneira simples

Git - Como gerenciar versões

- Git: sistema distribuído de controle de versões
- Criador: Linus Torvalds
- Permite a criação de diferentes versões do mesmo projeto (*branches*), que são compostas por modificações (*commits*) e que podem ser unidas (*merge*) para gerar uma nova versão
- GitHub: sistema em nuvem que permite o compartilhamento de repositórios Git
- Permite que usuários em diferentes lugares no mundo tenham acesso ao repositório
- Alternativas: GitLab, entre outras

Git - Configuração

- O Git é uma CLI instalada por padrão em qualquer distribuição Linux
- Para a utilização de forma passiva não necessita configuração (geralmente)
- Configurações básicas:
  - git --global config user.name “*User Name*”
  - git --global config user.email “*User Email*”
- Para repositórios públicos não necessita autenticação para clonar
- Para repositórios privados necessita autenticação via HTTPS ou SSH

Git - Autenticação via SSH

- É necessário um par de chaves
  - $ ssh-keygen -o
- A chave pública deve ser adicionada a conta do usuário
  - Settings -> SSH and GPG keys -> Add new SSH key
- Para acessar a chave pública pode usar o comando *cat*
  - $ cat ~/.ssh/id_rsa.pub
- Depois, basta copiar a saída do comando anterior e colar no GitHub
- Por fim, o projeto deve ser clonado usando a opção SSH

Clonar um repositório para o projeto

- Para melhor entendimento, vamos fazer um fork de repositório para o projeto exemplo
- Depois, vamos fazer alterações no projeto em outra máquina
- Essas alterações serão mandadas ao GitHub
- Essas alterações serão baixadas no servidor e será feita a atualização do projeto

Mão na Massa

- Agora, vamos instalar tudo o que for necessário e criar um projeto para colocá-lo em operação
- Serão utilizados:
  - Nginx
  - MySQL
  - Composer

Conclusão

- A lógica do processo utilizado serve para qualquer projeto em qualquer tecnologia
- O minicurso deve ser visto como um guia e não como um tutorial definitivo para qualquer projeto
- Cada projeto tem suas particularidades que devem ser estudadas e levadas em conta

Referências

https://99firms.com/blog/linux-statistics/

https://itsfoss.com/package-manager/

https://git-scm.com/

https://github.com

https://gitlab.com

Referências

https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-laravel-with-lemp-on-ubuntu-18-04

https://www.digitalocean.com/community/tutorials/how-to-install-and-configure-laravel-with-lemp-on-ubuntu-18-04

https://www.digitalocean.com/community/tutorials/how-to-install-linux-nginx-mysql-php-lemp-stack-ubuntu-18-04