CONFIGURAÇÃO DE AMBIENTE

AceleraDev Java

Oh my zsh (opcional)

Vamos instalar oh-my-zsh para que nosso bash fique bem parametrizado para que nosso ambiente fique com diversas funcionalidades e flexibilidade.

Para alterar do bash para o zsh basta rodar o comando chsh -s /bin/zsh ao encerrar a sessão e entrar novamente, o shell já estará mudado.

```
$ sh -c "$(wget
https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools
/install
.sh -O -)
$ source ~/.zshrc
$ chsh -s /bin/zsh
$ mkdir $HOME/go
```

Confira em seu arquivo passwd se está o path do zsh se não estiver coloque **“/bin/zsh”** `$ sudo vim /etc/passwd`

E adicionar as seguintes linhas no arquivo ~/.zshrc

```
$ export PATH=$PATH:/usr/local/go/bin
$ export GOPATH=$HOME/go
$ export PATH=$PATH:$GOPATH/bin
```

Exatamente como nós fizemos no profile. Caso ainda tenha dúvidas confere a instalação aqui.

Bash-it (opcional)

Uma alternativa ao oh-my-zsh para não sair do Bash é o bash-it. Ele traz algumas funcionalidades parecidas com o oh-my-zsh como prompt customizado (para mostrar informações do Git, por exemplo) e alguns alias. Para fazer sua instalação:

```
$ git clone --depth=1 https://github.com/Bash-it/bash-it.git
~/.bash_it #Para clonar o bash-it.
$ bash ~/.bash_it/install.sh
```

Ao rodar o script install.sh, o bash-it faz um backup automático do seu ~/.bashrc e adiciona algumas configurações no novo ~/.bashrc. Agora você pode editar o ~/.bashrc para customizar o seu bash-it. Mais informações podem ser encontradas no repositório oficial.

Git

Neste ambiente de desenvolvimento será utilizado a ferramenta de versionamento Git e o serviço de hospedagem de repositório GitHub.

Para instalação, acesse o site oficial do Git para fazer o download de acordo com o seu sistema operacional e siga o os passos de instalação. Case esteja utilizando Ubuntu, você pode simplesmente executar:

```
$ sudo apt update
$ sudo apt install git
```

Verifique se a instalação deu certo obtendo a versão do Git (a mais atual é a 2.21):

```
$ git version
```

Alguns links muito úteis para o aprendizado do Git são:

- ➔ Pro Git
- ➔ Atlassian Tutorials
- ➔ Plano para estudar Git e GitHub enquanto aprende programação
- ➔ Tudo que você queria saber sobre Git e GitHub, mas tinha vergonha de perguntar

GitHub

Para criar uma conta no GitHub, acesse o site oficial e preencha seus dados de nome de usuário, email e uma senha. Depois é só seguir os passos para verificação da criação da conta e você já estará pronto para utilizar o GitHub. Seu perfil será acessível pela URL https://github.com/<seu-nome-de-usuário>

Lembre-se de criar um nome de usuário que seja fácil de ler, lembrar e utilizar, pois outras pessoas (e até mesmo você) precisarão digitá-lo para trabalhar com seus projetos.

Para utilizar o GitHub através do protocolo SSH, você precisa configurar uma chave pública do SSH no site do GitHub. Para isso, siga os passos:

1. Primeiro verifique se na sua máquina já não existe um par de chaves SSH:

```
$ ls -al ~/.ssh
```

Se não houver os arquivos id_rsa.pub, id_dsa.pub ou algo similar é porque você ainda não possui um par de chaves SSH, então vamos criá-lo. Se já possui, vá para o passo 4.

2. Gere um par de chaves SSH (privada e pública) com criptografia RSA de 4096 bit. Utilize o mesmo email usado na criação da conta no GitHub:

```
$ ssh-keygen -t rsa -b 4096 -C “seu_email@dominio.com”
```

Quando perguntado onde deseja salvar o par de chaves, aceite a localização default (~/.ssh/id_rsa).

Quando perguntado por uma senha, digite uma senha segura e fácil de lembrar. Essa senha serve para desbloquear a sua chave SSH privada. Você precisará digitar essa senha quando interagindo com o GitHub pelo protocolo SSH.

3. Para diminuir o número de vezes que você precisará digitar essa senha, você pode adicioná-la ao agente do SSH:

```
$ eval “$(ssh-agent -s)”
$ ssh-add ~/.ssh/id_rsa
```

4. Agora vamos adicionar a sua chave pública do SSH no GitHub. Primeiro, copie o conteúdo da sua chave pública. Isso pode ser feito de diversas formas, por exemplo, abrindo o arquivo da chave pública, ~/.ssh/id_rsa.pub, selecionando **todo** seu conteúdo e copiando com Ctrl + C. Ou pela linha de comando:

```
$ cat ~/.ssh/id_rsa.pub | xclip -selection -c
```

Para adicionar a sua chave pública no GitHub, abra o site do GitHub, clique no seu ícone no canto superior direito e vá em Settings. No lado esquerdo, procure pela aba SSH and GPG Keys e depois clique no botão New SSH key.

No campo Title, dê um nome para a sua chave pública que identifique, por exemplo, o nome da máquina que você está trabalhando. Lembre-se, é possível trabalhar com GitHub a partir de várias máquinas e cada uma ter sua chave pública copiada para o GitHub.

No campo key, cole o conteúdo do arquivo ~/.ssh/id_rsa.pub que foi copiado anteriormente.

Clique em Add SSH key e preencha sua senha do GitHub, caso seja necessário.

5. Teste se tudo deu certo com:

```
$ ssh -T git@github.com
```

Se surgir uma mensagem de aviso perguntando se você deseja continuar, diga que sim (yes).

Pronto! Agora você deve ter total acesso à sua conta do GitHub utilizando o protocolo SSH.

Se estiver tendo problemas, consulte a documentação oficial ou peça ajuda a alguém da equipe Codenation.

Docker

Em nosso ambiente de desenvolvimento iremos utilizar o Docker para executar alguns serviços sem precisar instalá-los. É fortemente aconselhado utilizar Linux para nosso ambiente de desenvolvimento, pois isso torna muito mais fácil de configurar e gerenciar o Docker. Todas informações para a instalação do Docker podem ser obtidas no guia de instalação oficial.

Se estiver utilizando Ubuntu, você também pode seguir esses tutoriais: Como instalar e usar o Docker no Ubuntu 18.04 or Get Docker CE for Ubuntu.

Para facilitar no caso do Ubuntu, listamos os passos necessários abaixo:

```
$ sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key add -
$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"
$ sudo apt update
$ sudo apt install docker-ce docker-ce-cli containerd.io
$ apt-cache policy docker-ce
$ sudo systemctl status docker
$ sudo usermod -aG docker ${USER}
$ su - ${USER}
$ id -nG
$ sudo usermod -aG docker devel
```

Para descobrir se a instalação deu certo, execute a imagem de Hello World do Docker:

```
$ docker run hello-world
```

Você deve ver uma mensagem de boas-vindas do Docker.

Java

O próximo passo é instalar o Java 8.

Se estiver no ubuntu faça o download através deste link.

Instale utilizando o comando:

```
$ sudo dpkg --install
java-1.8.0-amazon-corretto-jdk_8.212.04-2_amd64.deb
```

No terminal utilize o comando:

```
$ java -version
```

A saída esperar será algo do tipo:

```
openjdk version "1.8.0_212"
OpenJDK Runtime Environment Corretto-8.212.04.3 (build 1.8.0_212-b04)
OpenJDK 64-Bit Server VM Corretto-8.212.04.3 (build 25.212-b04, mixed
mode)
```

Se, ao verificar a saída, estiver apontando para outra versão do java, utilize estes comandos para setar a versão correta:

```
$ sudo update-alternatives --config java
$ sudo update-alternatives --config javac
```

PostgreSQL

O próximo passo é instalar o PostgreSQL.

Limpe instalações anteriores:

```
$ sudo apt-get --purge -y remove postgresql\* && sudo apt-get -y autoremove
```

Instale o PostgreSQL e seus complementos:

```
$ sudo apt-get install -y postgresql-10
```

Defina a senha *postgres* para o usuário *postgres:*

```
$ sudo -u postgres psql -U postgres -d postgres -c "alter user postgres with password 'postgres';"
```

Reinicie o serviço:

```
$ sudo service postgresql reload
```

IntelliJ

1. IntelliJ
    a. Deve se entrar no site IntelliJ IDEA
        i. No momento esta documentação está utilizando mas recente do IntelliJ IDEA
    b. Realizar o download de acordo com seu sistema operacional
    c. Executar a instalação padrão