08

Conclusão

Transcrição

Parabéns! Você chegou agora ao final da segunda parte do curso de Shell Scripting. Vamos relembrar o que estudamos?

Começamos fazendo o script que realizava filtros em um log do servidor Apache. Só que precisávamos também fazer um tipo de validação para garantir que o parâmetro passado pelo usuário tinha aquele formato do endereço IPV4 com números e pontos. Então, nós criamos uma expressão regular para fazer uma validação maior do parâmetro que o usuário poderia passar.

Depois fizemos a parte do filtro baseado nas requisições `GET`, `POST`, `PUT` e `DELETE`. Utilizamos CASES, que é uma solução mais enxuta, mais elegante, onde ela validava todos os demais casos. Também tratamos o problema onde o usuário poderia colocar letras minúsculas, utilizando o comando `awk`, assim, nós transformamos todas as palavras em maiúsculas para ter sempre a garantia de que não teria nenhum problema caso o usuário passasse palavras em minúsculo.

Logo depois, os usuários precisavam acessar o conteúdo web do servidor Apache, e os diretores nos pediram para fazer uma monitoração desse servidor. Então, se o servidor caísse, deveríamos inicializá-lo e mandar um e-mail para o administrador do sistema, para que ele pudesse ver o que tinha acontecido. A estratégia que utilizamos para verificar se o serviço estava indisponível, era utilizar o **curl**, pois ele verifica o status HTTP. Se o status HTTP fosse diferente de 200, é porque não tínhamos mais o acesso ao conteúdo web, e então configuramos o envio do e-mail para o administrador da *Mutillidae*, e logo depois, reiniciávamos o servidor.

Mas, para alguém lembrar de rodar esse script toda hora não é muito bom, pois a chance de esquecerem é bem alta! Mas justamente por isso que nós utilizamos o **crontab**. O crontab agenda a execução de scripts em horários pré-determinados.

Depois, o cenário em questão era o da memória que atingia um determinado limiar, e quando esse consumo passava do limite, nós também devemos mandar um e-mail para o administrador para que ele pudesse saber o que estava acontecendo, passando também no e-mail, a informação do consumo em *gigabytes*.

No final, elaboramos um script que faz um *backup* do banco da *Mutillidae*, e vimos como poderíamos enviar essas informações para o bucket da Amazon, a seguir verificamos como poderíamos pegá-las de volta e colocar em uma tabela que apresentava algum problema. Para testar, nós deletamos a tabela `produtos`, e através do script `restore_amazon.sh`, conseguimos pegar o conteúdo do bucket na Amazon, e trouxemos para o diretório `/restore_mutillidae_amazon` com o comando `mysql -u root mutillidae`.

Esperamos te encontrar em uma próxima oportunidade!