**Powershell Parte 1 – Começando Com A Linguagem De Scpripting Moderna**

1. **Aula 1 – Apresentando o PowerShell:**
   1. Conhecendo o problema do cliente:
      1. Mstsc: Ao rodar no executar win+R, abre uma conexão de área remota.
      2. Podemos ver no gerenciador de tarefas o que está consumindo muito da máquina, clicar com o botão direito e ir para detalhes e a partir dos detalhes, clicar com botão direito e ir para serviços, assim descobrimos quais serviços estão utilizando aquela aplicação que está consumindo muito do nosso sistema.
      3. O problema do nosso cliente é exatamente esse, temos um serviço que está usando 90/100% da capacidade da cpu dos nossos servidores e precisamos reiniciar eles para que voltem ao normal, mas, conectar remotamente em um por um e fazer manualmente nos 10 servidores é muito trabalhoso, por isso vamos escrever um script que faça esse trabalho.
   2. Tentando solucionar com o antigo CMD:
      1. Criamos um script que identifica se um serviço está rodando ou não e se estiver faz reiniciar, se não estiver retorna que não está.
      2. O CMD não possui uma convenção de nomes definida. Ao contrário do PowerShell, o CMD não possui uma forte convenção de nomes.
      3. O CMD só trabalha com texto! Ou seja, para obtermos as informações que nos interessam em um script, muitas vezes, precisamos lidar com busca de palavras chaves no retorno dos comandos. Diferente do PowerShell, o CMD não é orientado a objetos.
      4. Muitas vezes é necessário conhecimento específico sobre os comandos que executamos, pois, cada um recebe argumentos e funciona de uma maneira diferente. O CMD está disponível até nas versões mais recentes do windows, como o Windows 10 ou o Windows Server 2016.
   3. Resolvendo o problema com PowerShell:
      1. Diferente do cmd, no powershell é tudo bem mais intuitivo, por exemplo, para obter os serviços vc digita exatamente isso Get-Service.
      2. Quando der enter o PS te devolve todos os serviços, parados e rodando, da sua máquina.
      3. Ao contrário do cmd que tínhamos que digitar sc query e afins, sem nem sequer sabermos para que serve ou que exatamente significa esse comando.
      4. Tudo no PS tem autocomplete, ou seja, se não souber o nome completo do comando, basta digitar uma parte dele e ir apertando tab até encontrar o que precisa.
      5. $variável = : Cria uma variável com seja lá o que você quiser. Pode inclusive obter um serviço e colocar dentro de uma variável:

$servico = Get-Service -Name "Themes"



* + - 1. Ao colocar um ‘.’ Depois da variável e ir dando tab podemos ver todas as funções que podemos utilizar com ela, como por exemplo:



* + 1. Ao invés de ficar reescrevendo todo o código redundantemente como no cmd, podemos fazer simplesmente assim:



* + 1. Já que o PS é orientado a objeto, tudo fica mais fácil e simples.

1. **Aula 2 – Usando Comandos Legados e Aprendendo Sem o Google:**
   1. Conhecendo o CommandPrecedence:
      1. Como questão de segurança, o PS faz com que todos os scipts do cmd escritos não rodem nele, mas, se realmente quiser rodar um script de cmd, precisa colocar ‘.\’ antes do .bat que quer executar, dessa forma ele irá rodar normalmente.
      2. Esse é o famoso commandPrecedence.
   2. Quando o CommandPrecedence não é necessário e o comando GetMember:
      1. Diferentemente do cmd, para podermos ver o nosso path, precisamos dar um echo na variável $env:path. Não só para o path, mas qualquer outra variável de ambiente que você queira. Basta substituir o path por ela, ou ir dando tab até encontrar.
      2. Para saber qual é o tipo de uma variável podemos colocar $env:variável.GetType().



* + 1. O path é sempre muito confuso principalmente quando está com muitos caminhos, mas, como podemos ver, ele é uma string portanto os métodos de string vão funcionar com ele também, e podemos utilizá-los para ter uma melhor exibição dele.
    2. Para procurar qualquer documentação na Microsoft sobre métodos ou suas linguagens de programação digite msdn e o que está procurando, como no caso de strings.
    3. O primeiro link nos leva pra cá: <https://docs.microsoft.com/en-us/dotnet/api/system.string?view=net-6.0>.
    4. Para deixar nosso path mais visível podemos usar o método split() de strings, passando como parâmetro o ‘;’ para fazer essa separação:



* + 1. Nem sempre teremos acesso a internet para poder pesquisar um método específico para trabalhar com nossa variável. Por conta disso podemos utilizar o comando Get-Member, que nos retorna uma lista de tudo o que podemos utilizar com nossa variável e suas descrições:

