SCP/SSH Powershell password

Установить модуль Posh-SSH

```
Import-Module PowershellGet
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned #Yes
Install-Module Posh-SSH #Yes
```

Скрипт бэкапа

```
Import-Module Posh-SSH

#list of adcs
$servers = @()
$servers += @{name = 'adc01'; ip = '192.168.124.11'}
$servers += @{name = 'adc02'; ip = '192.168.124.12'}



##touch this:
#username:
$nsUser = "nsroot"

#path to file contains encrypted pass:
$credPath = "./adcpasswd.txt"



$backupStorePath = "./"
$doFullBackup = $true #or $false $true - make full, $false - make basic
$eraseBackup = $false #or $true  $true - erase backup from adc after
transfer, $false - leave backup on adc


##ask for creds if password file does not exist
if ($(Test-Path -Path $credPath) -ne $true) {
    $cred = Get-Credential -Credential $nsUser
    $cred.Password | ConvertFrom-SecureString | Set-Content $credPath
}

#do not touch that:
$backup_level = "basic", "full"
$encryptedPass = Get-Content $credPath | ConvertTo-SecureString
$cred = New-Object System.Management.Automation.PsCredential($nsUser,
```

```
$encryptedPass)

function remove_backup {
    param (
        $server,
        $backupName
    )
    $sshCommand = "rm system backup " + $backupName +".tgz"
    Write-Host($sshCommand)
    $sshSession = New-SSHSession -ComputerName $server.ip -Credential $cred
-AcceptKey -Force
    $result = Invoke-SSHCommand -Command $sshCommand -SSHSession $sshSession
    if ($result.ExitStatus -eq 0) {
            Write-Host("remove: ",$server.name,$backupname)
    }
    else {
        Write-Host($result.Host, $result.Output)
    }
    Remove-SSHSession -SessionId $sshSession.SessionId
}

function get_backup {
    param (
        $server,
        $backupName,
        $destinationPath = "./",
        $erase
    )
    $backup_file_path = "/var/ns_sys_backup/$backupName.tgz"
    Write-Host($backup_file_path)
    $result = Get-SCPItem `
        -ComputerName $server.ip `
        -Credential $cred `
        -Path $backup_file_path `
        -PathType File `
        -Destination $destinationPath `
        -Force
    if ($erase -eq $true){remove_backup -server $server -backupName
$backupName}
    return $result
}

function create_backup {
    param (
        $server,
        $doFullBackup = $true,
        $erase = $false
    )

    $backup_name_pattern = $server.name+ "_" + $backup_level[$doFullBackup]
+ "_" + $(get-date -format "yyyy_MM_dd")
```

```
    Write-
Host($server.name,$backup_name_pattern,$backup_level[$doFullBackup], $erase)
    $sshCommands = @()
    $sshCommands += "set HA node -haSync DISABLED -haProp DISABLED" #0
    $sshCommands += "save ns config" #1
    $sshCommands += "create system backup " + $backup_name_pattern +" -level
" + $backup_level[$doFullBackup] #2
    $sshCommands += "set HA node -haSync ENABLED -haProp ENABLED" #3
    $sshCommands += "save ns config" #4
    $sshSession = New-SSHSession -ComputerName $server.ip -Credential $cred
-AcceptKey
    Invoke-SSHCommand -Command $sshCommands[0] -SSHSession $sshSession
    Invoke-SSHCommand -Command $sshCommands[1] -SSHSession $sshSession
    $result = Invoke-SSHCommand -Command $sshCommands[2] -SSHSession
$sshSession
    if ($result.ExitStatus -eq 0) {
        $result = get_backup -server $server -backupName
$backup_name_pattern -destinationPath $backupStorePath -erase $erase
    }
    else {
        Write-Host($result.Host, $result.Output)
    }
    Invoke-SSHCommand -Command $sshCommands[3] -SSHSession $sshSession -
    Invoke-SSHCommand -Command $sshCommands[4] -SSHSession $sshSession
    Remove-SSHSession -SessionId $sshSession.SessionId
}



foreach ($server in $servers) {
    create_backup -server $server -doFullBackup $doFullBackup -erase
$eraseBackup
}
```