Módulo SSH

O módulo **ssh** fornece funções para executar comandos remotos em servidores através do protocolo Secure Shell (SSH). Oferece duas abordagens principais: execução única de comandos e conexões persistentes para múltiplos comandos. O módulo suporta autenticação por senha e inclui resolução DNS automática.

Funções Disponíveis

1. `ssh.exec(opts)`

Executa um comando remoto via SSH de forma assíncrona e retorna a saída do comando.

Parâmetros:

- `opts` (tabela): Opções de conexão e execução contendo:
  - `host` (string): Endereço do servidor (hostname ou IP)
  - `username` (string): Nome de usuário para autenticação
  - `password` (string): Senha para autenticação
  - `command` (string): Comando a ser executado no servidor remoto
  - `port` (número opcional, padrão: 22): Porta SSH
  - `timeout` (número opcional, padrão: 60): Timeout em segundos para a conexão

Retorno:

- `string`: Saída padrão (stdout) do comando executado

Exceções:

- Lança erro se algum parâmetro obrigatório estiver faltando
- Lança erro se a conexão SSH falhar
- Lança erro se o comando falhar no servidor remoto
- Lança erro "connection timeout" se exceder o tempo limite

Exemplos:

```
-- Execução básica de comando
local opts = {
```

```
    host = "servidor.exemplo.com",
    username = "admin",
    password = "senha123",
    command = "uname -a",
    port = 22,
    timeout = 30
}

local success, output = pcall(function()
    return ssh.exec(opts)
end)

if success then
    print("Saída do comando: " .. output)
else
    print("Erro SSH: " .. output)
end

-- Verificar uso de disco
local disk_opts = {
    host = "192.168.1.100",
    username = "monitor",
    password = "monitor_pass",
    command = "df -h /"
}

local disk_usage = ssh.exec(disk_opts)

-- Verificar serviços em execução
local service_opts = {
    host = "app-server",
    username = "root",
    password = "root_password",
    command = "systemctl list-units --type=service --state=running"
}

local services = ssh.exec(service_opts)
```

2. `ssh.connect(opts)`

Estabelece uma conexão SSH persistente e retorna um objeto cliente que pode executar múltiplos comandos.

Nota: Esta função usa valores padrão da tabela `params` quando os parâmetros não são fornecidos.

Parâmetros:

- `opts` (tabela): Opções de conexão contendo:
  - `host` (string): Endereço do servidor (hostname ou IP)
  - `username` (string): Nome de usuário para autenticação
  - `password` (string): Senha para autenticação
  - `port` (número opcional, padrão: 22): Porta SSH
  - `timeout` (número opcional, padrão: `params.sshTimeout` ou 60): Timeout em segundos para a conexão

Retorno:

- `SshClient`: Objeto cliente SSH com método `exec()` para executar comandos

Exceções:

- Lança erro se algum parâmetro obrigatório estiver faltando
- Lança erro se a conexão SSH falhar
- Lança erro "connection timeout" se exceder o tempo limite

Exemplos:

```
-- Criar conexão persistente
local connect_opts = {
    host = "banco-de-dados.exemplo.com",
    username = "dba",
    password = "dba_password",
    port = 2222,
    timeout = 45
}

local success, client = pcall(function()
    return ssh.connect(connect_opts)
end)

if not success then
    print("Falha na conexão SSH: " .. client)
    return
```

```
end

-- Executar múltiplos comandos na mesma conexão
local cmd1_output = client:exec("pg_isready")
local cmd2_output = client:exec("psql -c 'SELECT version();'")
local cmd3_output = client:exec("df -h /var/lib/postgresql")

print("PostgreSQL status: " .. cmd1_output)
print("Versão PostgreSQL: " .. cmd2_output)
print("Espaço em disco: " .. cmd3_output)
```

3. `SshClient:exec(command)`

Método do objeto cliente retornado por `ssh.connect()` para executar comandos na conexão estabelecida.

Parâmetros:

- `command` (string): Comando a ser executado no servidor remoto

Retorno:

- `string`: Saída padrão (stdout) do comando executado

Exceções:

- Lança erro se a conexão subjacente falhar
- Lança erro se o comando falhar no servidor remoto

Exemplos:

```
-- Uso com múltiplos comandos relacionados
-- Exemplo usando valores explícitos
local client = ssh.connect({
    host = "monitor-server",
    username = "metrics",
    password = "metrics_pass"
})

-- Exemplo usando valores do params configurado no dispositivo
-- params.sshUsername = "admin"
```

```
-- params.sshPassword = "admin123"
-- params.address = "servidor.local"
-- params.sshPort = 22
-- params.sshTimeout = 30

local client_simplificado = ssh.connect({})
-- Equivalente a: ssh.connect({
--     host = "servidor.local",
--     username = "admin",
--     password = "admin123",
--     port = 22,
--     timeout = 30
-- })

-- Exemplo misto (alguns valores explícitos, outros do params)
local client_misto = ssh.connect({
    host = "backup-server",
    port = 2222
})
-- Usa: host = "backup-server", port = 2222
-- Usa do params: username = "admin", password = "admin123", timeout = 30

-- Coletar métricas do sistema
local uptime = client:exec("uptime")
local memory = client:exec("free -m")
local cpu = client:exec("top -bn1 | grep 'Cpu(s)'")
local disk = client:exec("iostat -x 1 1")

-- Processar resultados
print("Uptime: " .. uptime)
print("Memória: " .. memory)
print("CPU: " .. cpu)
print("Disk I/O: " .. disk)
```

Informações Adicionais

1. **Resolução DNS Automática**

- Hostnames são automaticamente resolvidos para endereços IP
- Usa o módulo DNS interno do Monsta
- Suporta tanto IPv4 quanto IPv6

2. **Conexões Persistentes**

- Conexões SSH podem ser reutilizadas para múltiplos comandos
- Reduz overhead de autenticação para comandos sequenciais
- Melhora performance em operações em lote

3. **Timeout Configurável**

- Timeout padrão de 60 segundos para conexões
- Configurável por conexão via parâmetro `timeout`
- Prevenção contra conexões travadas

Limitações

1. **Autenticação Apenas por Senha**

- Não suporta autenticação por chave pública (SSH key)

2. **Performance**

- Conexões SSH têm overhead significativo
- Não recomendado para comandos muito frequentes (use agentes locais)

Revision #1
Created 3 February 2026 16:45:21 by Marc
Updated 3 February 2026 16:45:33 by Marc