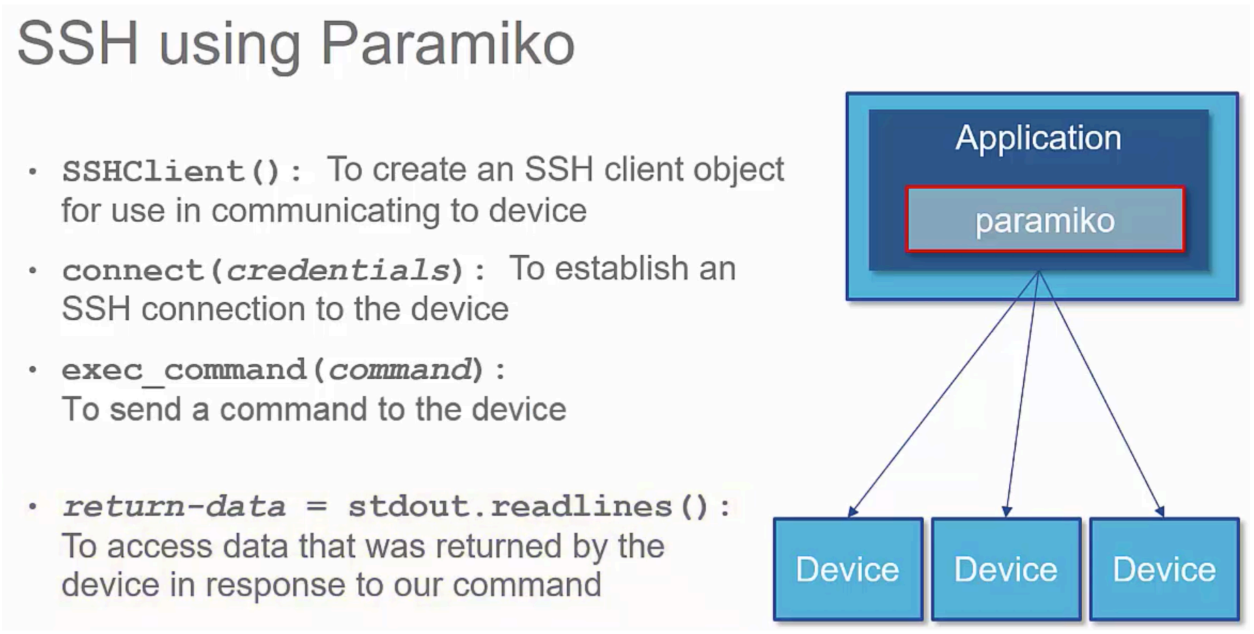
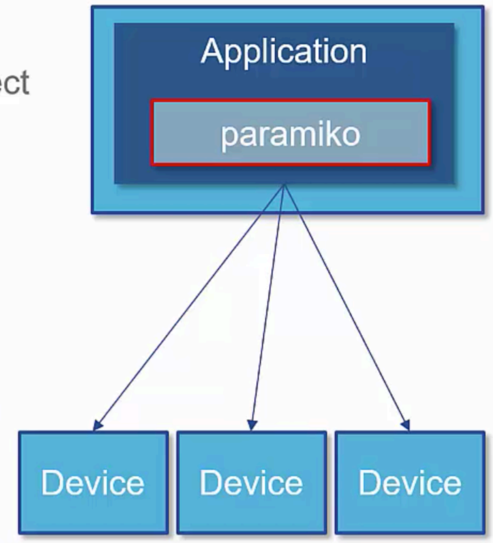
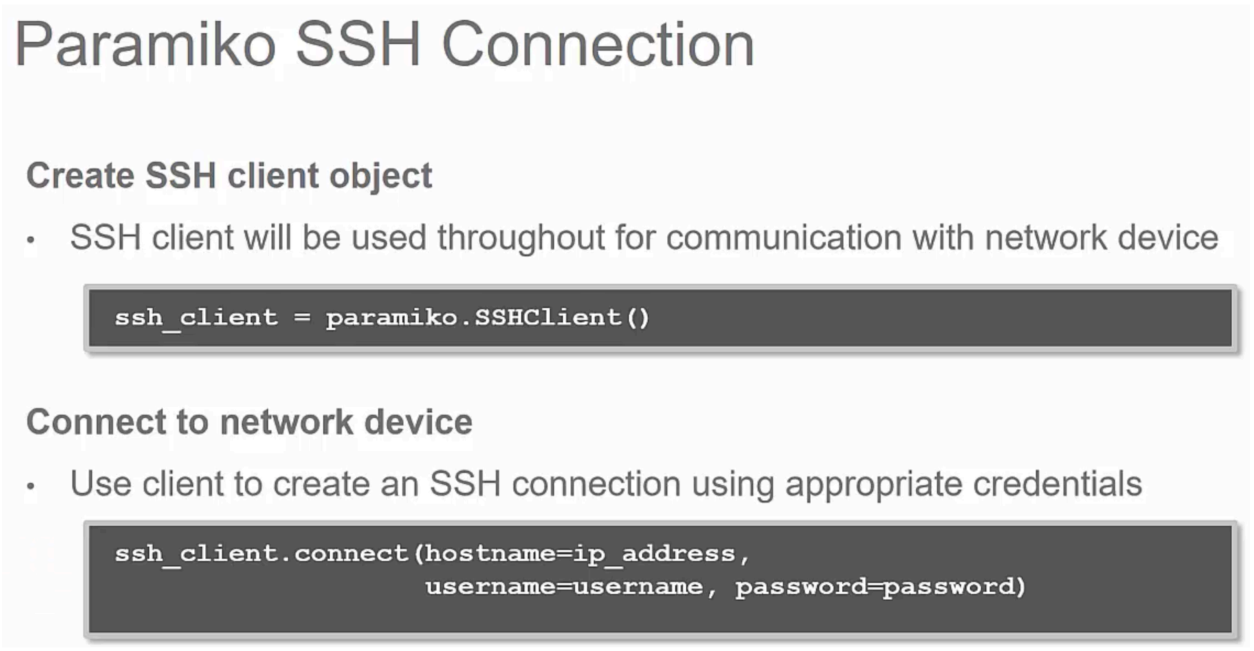


SSH using Paramiko

- `SSHClient():` To create an SSH client object for use in communicating to device
- `connect(credentials):` To establish an SSH connection to the device
- `exec_command(command):` To send a command to the device
- `return-data = stdout.readlines():` To access data that was returned by the device in response to our command





Paramiko SSH Connection

Create SSH client object

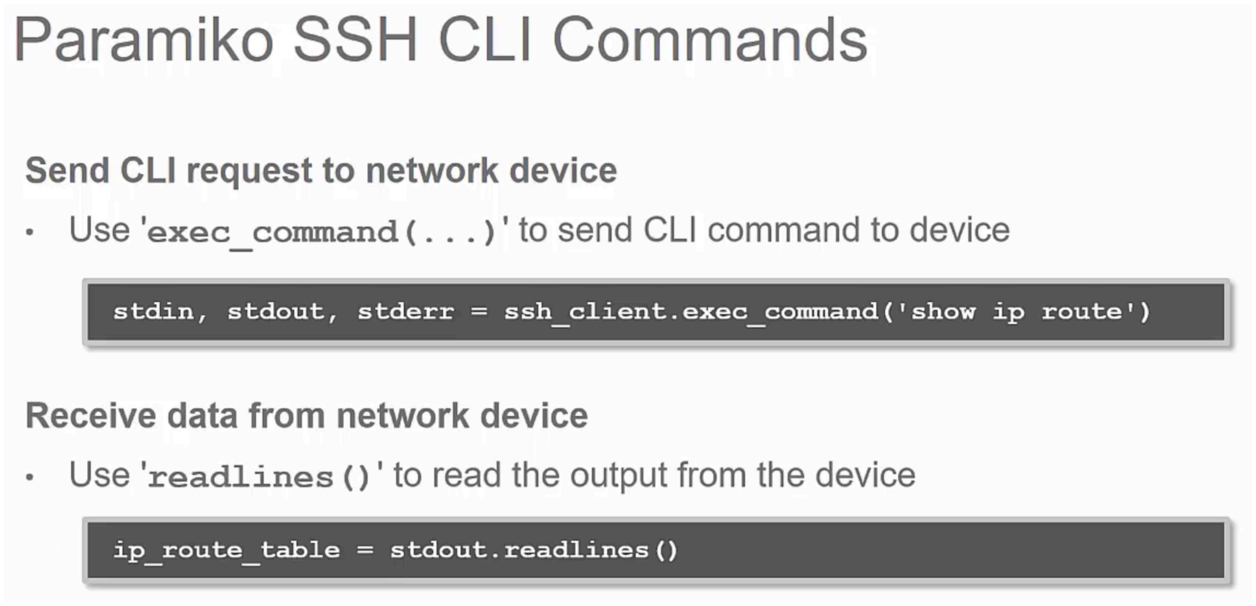
- SSH client will be used throughout for communication with network device

```
ssh_client = paramiko.SSHClient()
```

Connect to network device

- Use client to create an SSH connection using appropriate credentials

```
ssh_client.connect(hostname=ip_address,
                   username=username, password=password)
```



Paramiko SSH CLI Commands

Send CLI request to network device

- Use '`exec_command(...)`' to send CLI command to device

```
stdin, stdout, stderr = ssh_client.exec_command('show ip route')
```

Receive data from network device

- Use '`readlines()`' to read the output from the device

```
ip_route_table = stdout.readlines()
```

Connecting to a device using SSH using Pexpect is very similar to connecting via Telnet. The major difference is in the command that is used when you launch the session.

Connect Via SSH Using Pexpect

To create ('spawn') a Pexpect session using the pexpect.spawn(...) method:

```
# Create pexect SSH session
session = pexpect.spawn('ssh ' + username + '@' + ip_address, timeout=20)
```

The 'spawn' method creates a new session, just as with telnet, giving you a session for sending commands, receiving and parsing output, and so on. The rest of the login process is the same as for telnet, with the exception being that there is no need to have code to enter the 'username', since that has already been provided in the SSH command line itself.

Connect Via SSH Using Paramiko

Paramiko is a Python implementation of SSHv2 that provides both client and server functionality. The following is an example of establishing an SSH connection to a device using Paramiko:

```
import paramiko

ip_address = '10.30.30.1'
username = 'cisco'
password = 'cisco'

print '\n------------------------------------------------------'
print '--- Attempting paramiko connection to: ', ip_address

# Create paramiko session
ssh_client = paramiko.SSHClient()

# Must set missing host key policy since we don't have the SSH key
# stored in the 'known_hosts' file
ssh_client.set_missing_host_key_policy(paramiko.AutoAddPolicy())

# Make the connection to our host.
ssh_client.connect(hostname=ip_address,
                   username=username,
                   password=password)
```

```
# If there is an issue, paramiko will throw an exception,
# so the SSH request must have succeeded.

print '--- Success! connecting to: ', ip_address
print '---               Username: ', username
print '---               Password: ', password
print '------------------------------------------------------\n'
```

Once the SSH connection is established, commands can be executed using the `ssh_client.exec_command` function which uses three variables: `stdin`, `stdout`, and `stderr`. `stdin` passes the command to the device, `stdout` is used to store the response, and `stderr` is used to report any errors. For example, to obtain the IP route table from a router, you would enter:

```
stdin, stdout, stderr = ssh_client.exec_command('show ip route')
ip_route_table = stdout.readlines()

ssh_client.close()
```

To close the SSH session use `ssh_client.close()`

** Make sure the device is setup with SSH!

```
r1>en
Password: cisco
r1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
r1(config)#ip domain-name cisco.com
r1(config)#crypto key generate rsa
The name for the keys will be r1.cisco.com
Choose the size of the key modulus in the range of 360 to 4096 for your
  General Purpose Keys.  Choosing a key modulus greater than 512 may take
  a few minutes.

How many bits in the modulus [512]: 1024
Generating 1024 bit RSA keys, keys will be non-exportable...
[OK] (elapsed time was 0 seconds)

r1(config)#
* Apr 12 13:52:35.463: %SSH-5-ENABLED: SSH 1.99 has been enabled
r1(config)#ip ssh version 2
r1(config)#ip ssh rsa keypair-name r1.cisco.com
r1(config)#
* Apr 12 13:54:11.919: %SSH-5-DISABLED: SSH 2.0 has been disabled
* Apr 12 13:52:41.337: %SSH-5-ENABLED: SSH 2.0 has been enabled
r1(config)# end
r1#
```