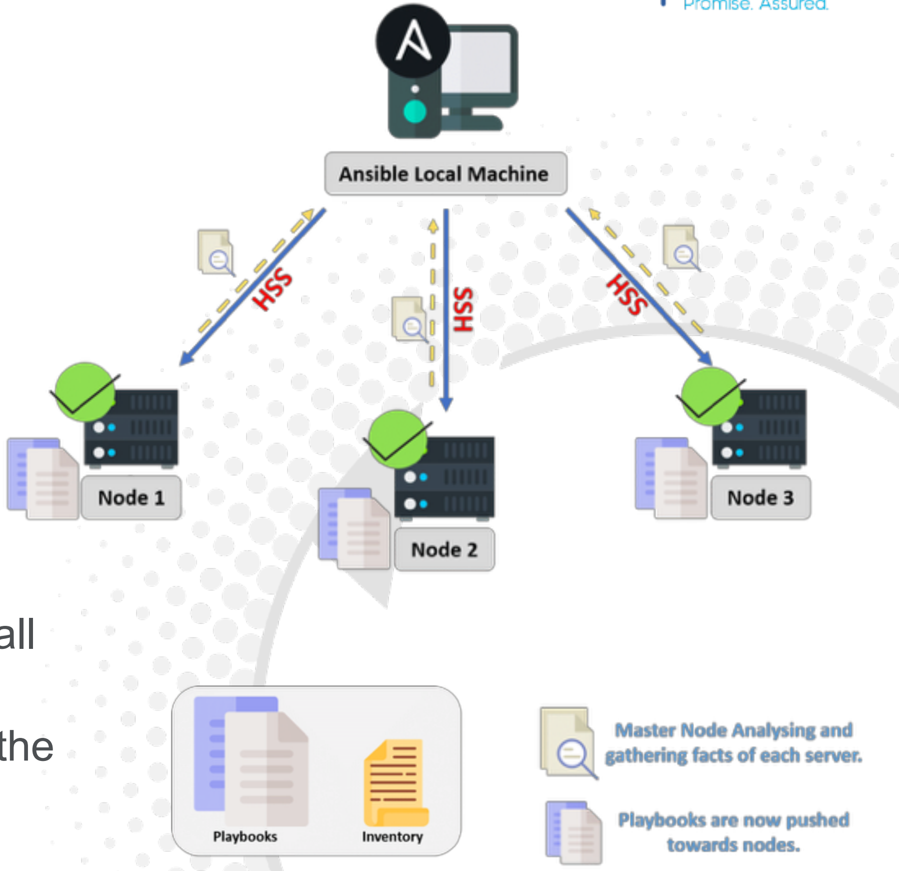
STC & ANSIBLE

Data-Modeling using ansible Automation

Version 1.0
2020-JAN-16th

What is Ansible

- Primary use-case for sensible
  - Configuring a large cluster of servers
  - Example: Installing docker on 1k servers
  - **It's all about configuring servers at scale**
- Terminology
  - **Playbook**: a list of sensible tasks which sensible execute sequential. Example: "install docker; add user spirent"
  - **Inventory**: A list of target (*nodes*) on which the playbook has to be executed.


Ansible Local Machine
SSH
SSH
SSH
Node 1
Node 2
Node 3
Playbooks
Inventory
Master Node Analysing and gathering facts of each server.
Playbooks are now pushed towards nodes.

Ansible and STC?

- Use Cases
  - **Ansible:** It's all about server config
  - **STC:** It's all about data-model config
- Examples
  - Create a PPPoE client & server, bind the client and wait for IP to be learned.
  - Example 2: Create a stream network mesh between 100 ports & generate traffic

Terminology

- **Inventory**:
  - List of chassis
  - List of lab-servers (usually, only one)
- **Playbook**:
  - List of tasks to create a data-model: eg "create", "config", "perform" etc

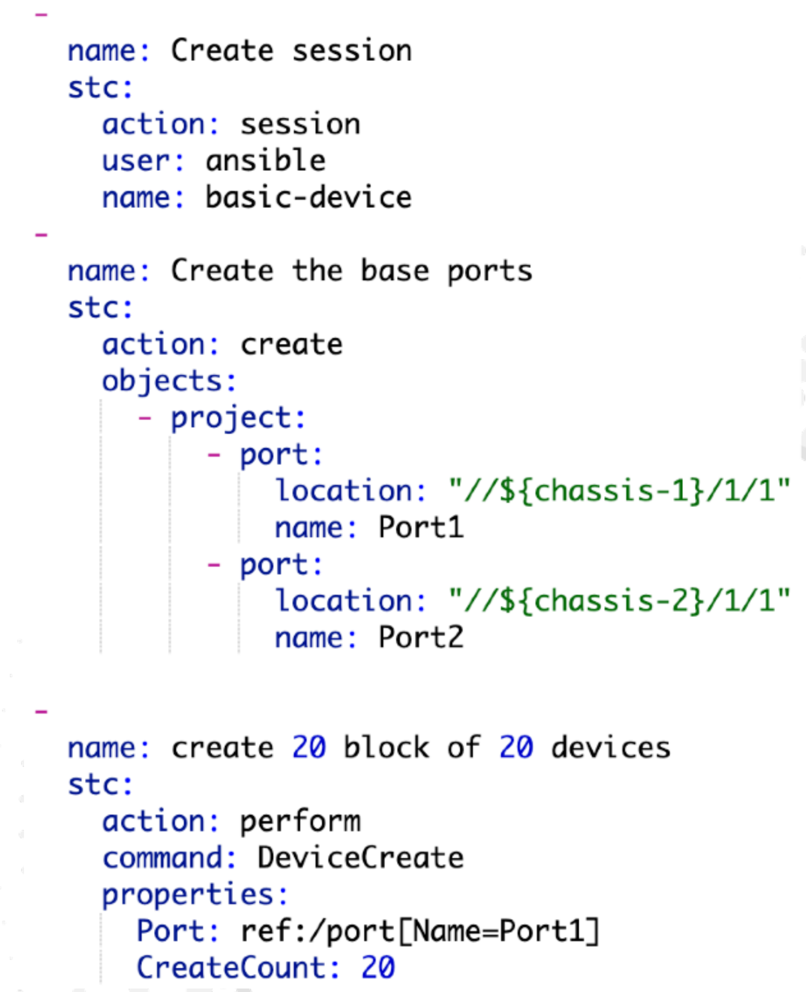
Ansible Playbooks

```yaml
tasks:
 -
  name: "add cache dir"
  file:
      path: /opt/cache
      state: directory

 -
   name: "install nginx"
   yum:
      name: nginx
      state: latest

 -
  name: "restart nginx"
  service:
      name: nginx
      state: restarted
```

- Using the “**yaml**” syntax
  - http://www.yamllint.com/
- Each task in the playbook
  - **name**: arbitrary description
  - **module**: eg “file”, “yum”, “service”
- Task properties are module specific:
  - “path” is only valid of “file”
  - “name” is valid for “yum” and “service”

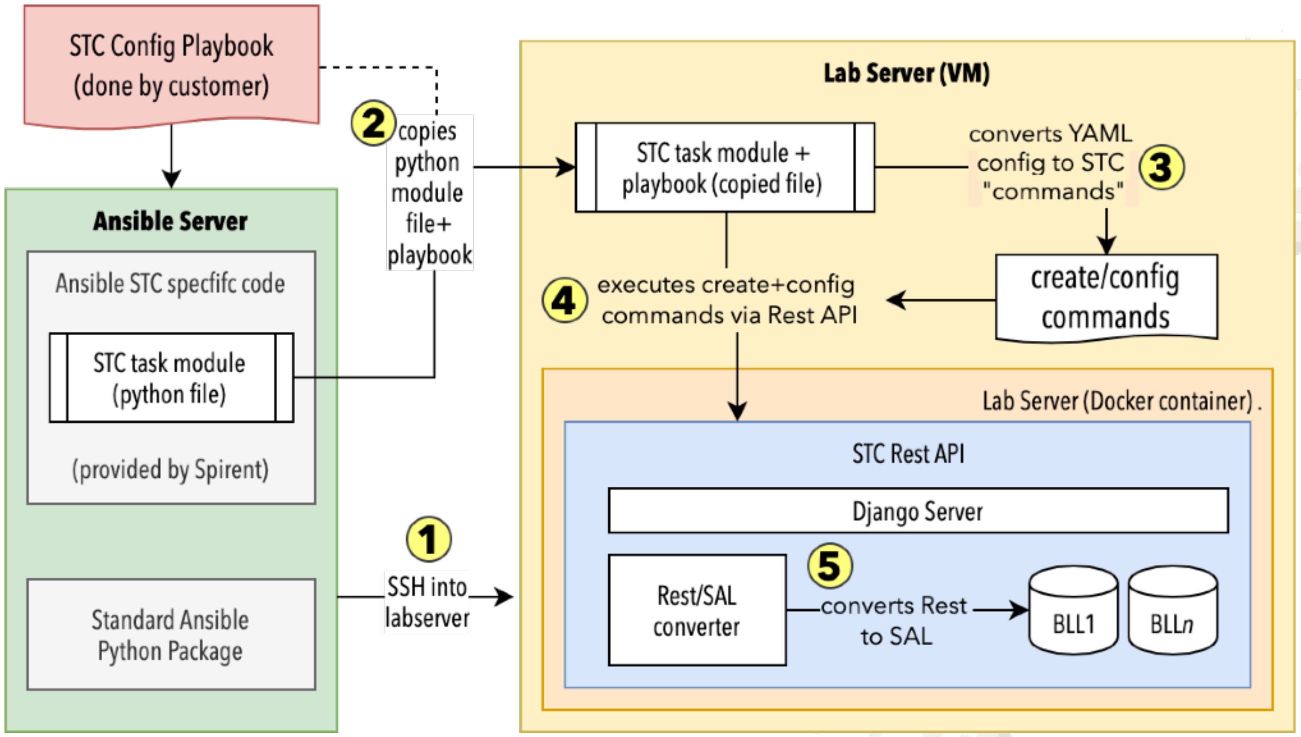
Ansible Playbooks … for STC

- How to support STC for sensible
  - Create an “**stc ansible module**”
  - This module “talks” to the lab-server
- The stc module can handle 8 tasks:
  - Session
  - Config, Create, Perform
  - Load (data-model)
  - Get, Wait
  - Download



```
-
  name: Create session
  stc:
    action: session
    user: ansible
    name: basic-device
-
  name: Create the base ports
  stc:
    action: create
    objects:
      - project:
        - port:
            location: "//${chassis-1}/1/1"
            name: Port1
        - port:
            location: "//${chassis-2}/1/1"
            name: Port2

-
  name: create 20 block of 20 devices
  stc:
    action: perform
    command: DeviceCreate
    properties:
      Port: ref:/port[Name=Port1]
      CreateCount: 20
```

How Ansible controls the Lab-Server?


STC Config Playbook (done by customer)
Ansible Server
Ansible STC specfifc code
STC task module (python file)
(provided by Spirent)
Standard Ansible Python Package
2 copies python module file+ playbook
1 SSH into labserver
Lab Server (VM)
STC task module + playbook (copied file)
converts YAML config to STC "commands" 3
create/config commands
4 executes create+config commands via Rest API
Lab Server (Docker container) .
STC Rest API
Django Server
Rest/SAL converter
5 converts Rest to SAL
BLL1
BLLn

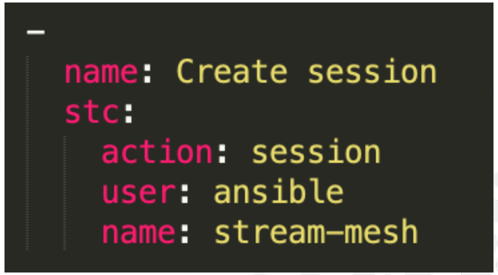
An STC playbook Example

Ansible and STC: Playbook example

- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic

Ansible and STC: Playbook example

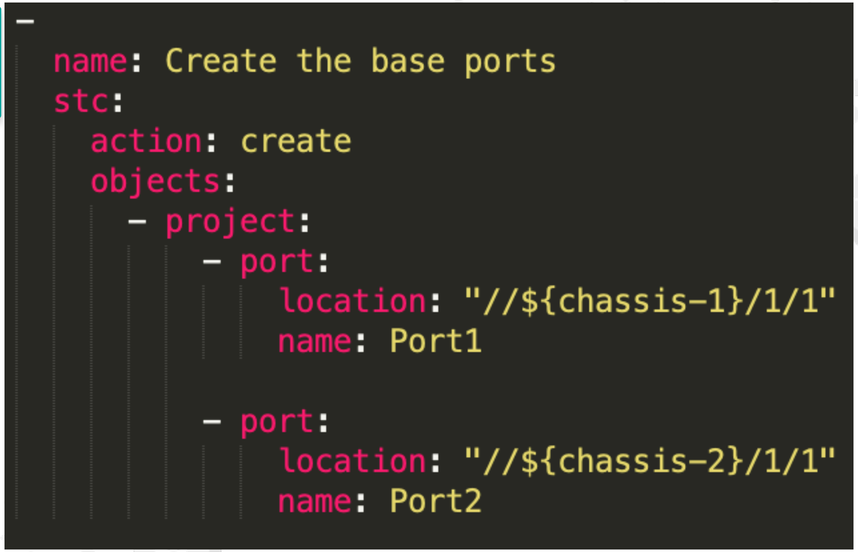
- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic



```
-
  name: Create session
  stc:
    action: session
    user: ansible
    name: stream-mesh
```

Ansible and STC: Playbook example

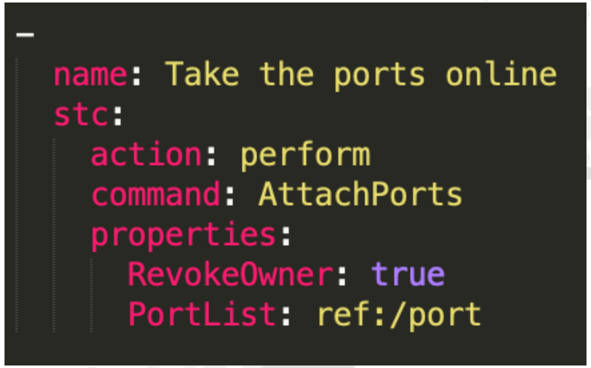
- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic



```
-
  name: Create the base ports
  stc:
    action: create
    objects:
      - project:
        - port:
            location: "//${chassis-1}/1/1"
            name: Port1

        - port:
            location: "//${chassis-2}/1/1"
            name: Port2
```

Ansible and STC: Playbook example

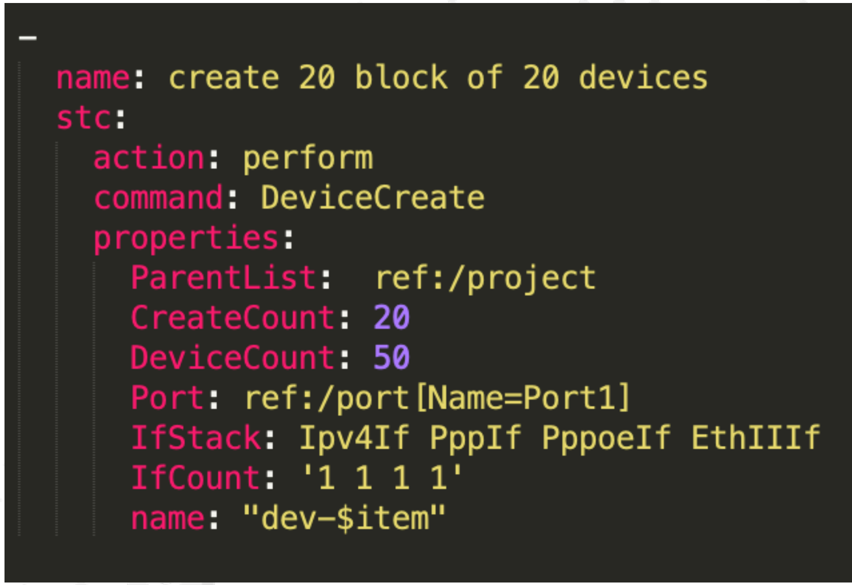
- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic



```
-
  name: Take the ports online
  stc:
    action: perform
    command: AttachPorts
    properties:
      RevokeOwner: true
      PortList: ref:/port
```

Ansible and STC: Playbook example

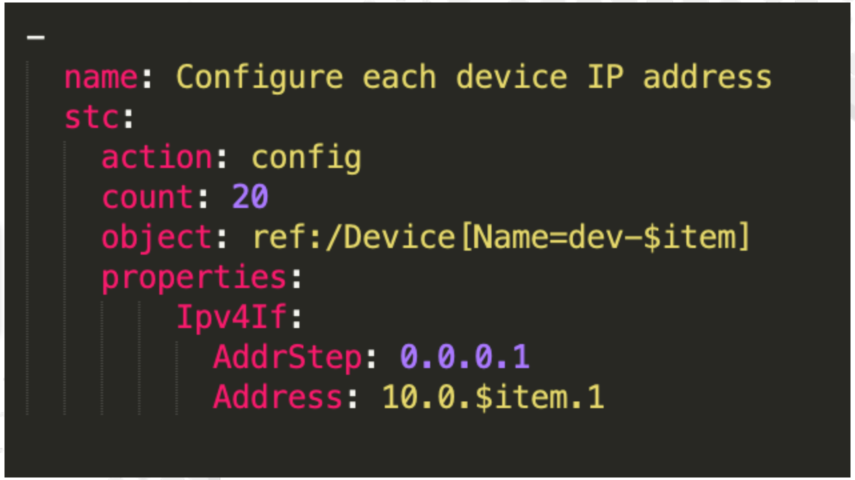
- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic



```
-
  name: create 20 block of 20 devices
  stc:
    action: perform
    command: DeviceCreate
    properties:
      ParentList:  ref:/project
      CreateCount: 20
      DeviceCount: 50
      Port: ref:/port[Name=Port1]
      IfStack: Ipv4If PppIf PppoeIf EthIIIf
      IfCount: '1 1 1 1'
      name: "dev-$item"
```

Ansible and STC: Playbook example

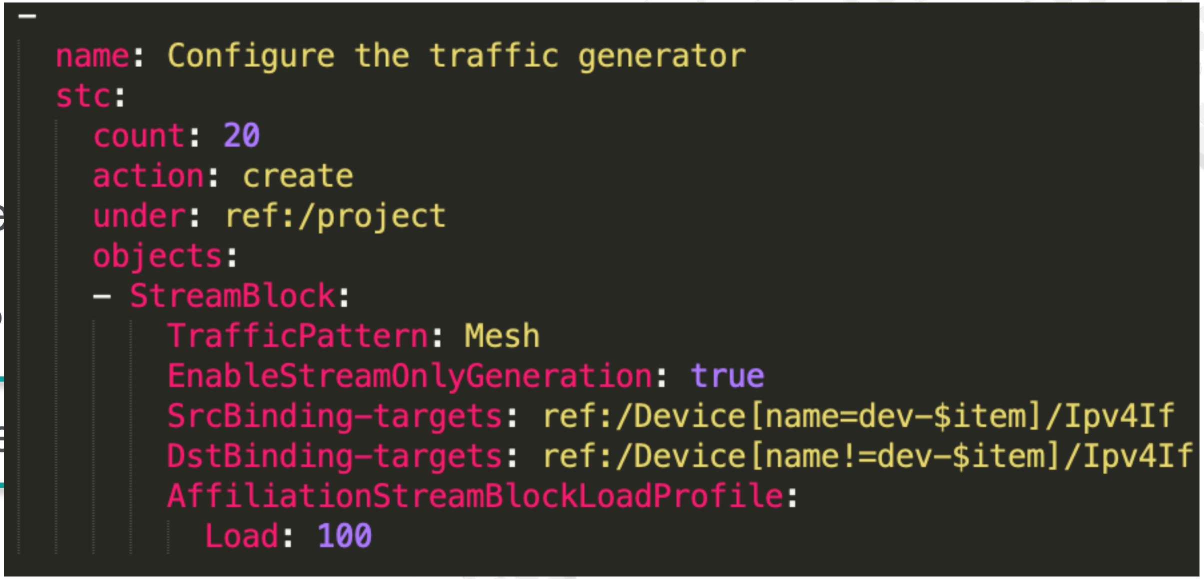
- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic



```
-
  name: Configure each device IP address
  stc:
    action: config
    count: 20
    object: ref:/Device[Name=dev-$item]
    properties:
        Ipv4If:
          AddrStep: 0.0.0.1
          Address: 10.0.$item.1
```

Ansible and STC: Playbook example

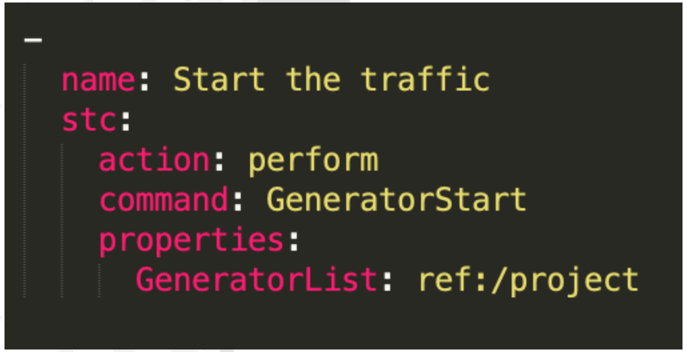
- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device
- **configure**: each device’s IP
- **create**: stream blocks betwe
- **perform**: start the traffic



```
-
  name: Configure the traffic generator
  stc:
    count: 20
    action: create
    under: ref:/project
    objects:
    - StreamBlock:
        TrafficPattern: Mesh
        EnableStreamOnlyGeneration: true
        SrcBinding-targets: ref:/Device[name=dev-$item]/Ipv4If
        DstBinding-targets: ref:/Device[name!=dev-$item]/Ipv4If
        AffiliationStreamBlockLoadProfile:
          Load: 100
```

Ansible and STC: Playbook example

- **session**: create a new session
- **create:** 2 “ports”
- **perform**: take ports online
- **create**: 20 “emulated device”
- **configure**: each device’s IP address
- **create**: stream blocks between each device
- **perform**: start the traffic

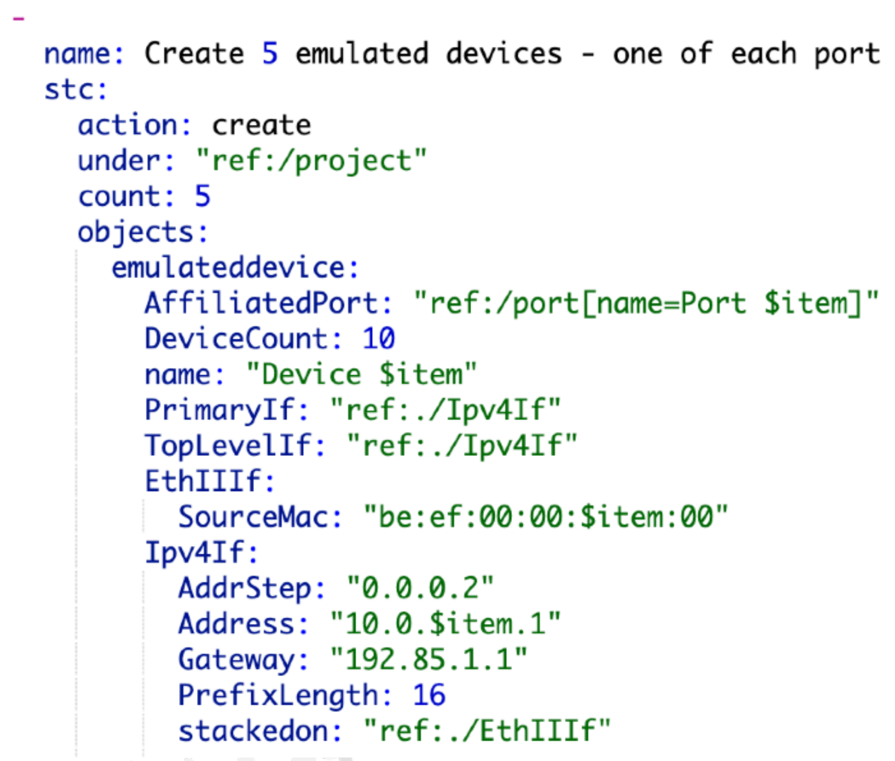


```yaml
-
  name: Start the traffic
  stc:
    action: perform
    command: GeneratorStart
    properties:
      GeneratorList: ref:/project
```

Advanced Concepts

References

- An X-Path like selector
  - ref:/project - (instead of “project1”)
  - ref:/port
  - ref:/port[name=Port 1]
  - ref:/port[0]
  - ref:/Device[name!=device-1]
  - ref:/Device[name!=device-1]/Ipv4If
  - ref:./Ipv4If



```
-
  name: Create 5 emulated devices - one of each port
  stc:
    action: create
    under: "ref:/project"
    count: 5
    objects:
      emulateddevice:
        AffiliatedPort: "ref:/port[name=Port $item]"
        DeviceCount: 10
        name: "Device $item"
        PrimaryIf: "ref:./Ipv4If"
        TopLevelIf: "ref:./Ipv4If"
        EthIIIf:
          SourceMac: "be:ef:00:00:$item:00"
        Ipv4If:
          AddrStep: "0.0.0.2"
          Address: "10.0.$item.1"
          Gateway: "192.85.1.1"
          PrefixLength: 16
          stackedon: "ref:./EthIIIf"
```

Iterators & Templates

- Purpose
  - Make it possible to configure more than one object in a task
  - Example: create 5 ports, create 100 emulated devices
- How is it done
  - Using the property “count” (eg count: 100) or “range” (eg range: A B C)
  - The tasks are templates, and “item” is the iteration counter
  - Anything encloses with “${…}” is a python expression

```
name: Create 5 base ports
stc:
  action: create
  count: 100
  objects:
    project:
      port:
        location: "//(Offline)/${item%8}/${item/8}"
        name: "Port $item"
```

```
name: create 20 block of 20 devices
stc:
  action: perform
  command: DeviceCreate
  properties:
    ParentList:  "ref:/project"
    CreateCount: 20
    DeviceCount: 50
    Port: "ref:/port[Name=Port1]"
    IfStack: "Ipv4If PppIf PppoeIf EthIIIf"
    IfCount: '1 1 1 1'
    name: "dev-$item"
```