Automation Myths

Automation

This is one of the coolest words nowadays, in the networking and security world. I believe this is due to the general idea that 'SDN' inherently implies an embedded automation, which can even be improved with some manual intervention. There is something true behind this idea, for example if we think about SD-WAN: with relatively low effort, you can add a new branch site for a customer, without having to buy a dedicated firewall to manage the backup traffic through the internet configuring an IpSec tunnel, with all the Qos and routing problems that this approach implies. For people working in the networking area since many years, there's a general feeling that you loose control over the way traffic is handled, thus potentially making troubleshooting a nightmare. This could be even more the feeling for SDN solutions applied to Service Providers backbones, along with Segment Routing and traffic engineering solutions based on a PCE external device. Commercial products seem to be more mature as years pass by, and for my personal experience that feeling didn't find so many real life proofs, but is not yet gone away.

One general misconception in my opinion is that 'a central brain' is something mandatory to provide 'automation', probably because this is the way SDN solutions were sold in the beginning: a centralized brain with the full vision of all the 'stupid' network elements and fully controls them, thus providing a better service for the traffic, than what a single network element could ever provide. But this is **absolutely not true**: automation and network configuration through scripting languages connecting many devices potentially performing complex network configurations didn't' need a centralized brain, they just needed programming skills possibly coupled with networking skills. A couple of things that rarely go together, even though this seems to be the only direction for the future, if someone still want to work as a network engineer (aka learning Python is no more optional nowadays). ACI by Cisco is a clear example: automating the leaf/spines day-1 configurations given a couple of aggregated private spaces for unicast and multicast addresses, was something doable with a Python script, just like configurations to stretch a vlan between a couple of DataCenters connected with OTV technology. What has been **fundamental** from the automation point of view, was the introduction of the **REST API** standard to interact with ANY possible system, being it a router, a firewall, a load balancer or a vSphere cluster for the management of a virtualized (network) environment. Not every vendor was providing the possibility to configure their systems through a CLI (Command Line Interface), some of them only provided a GUI. This new standard approach has been widely accepted and no vendor can nowadays avoid supporting it, even though from the number of tech blogs, github projects, bugs and problems, this approach to configure devices is not a highway, but looks more like a steepy unpaved road. REST API requests have standardized the transport protocol (http/https), but of course every vendor has its own 'objects' to be queried (better or worse documented …) to retrieve data to gather useful information, or push data to configure the device or the devices controlled by the central brain or management system (like Cisco EPNM). Another aspect is the way information are organized when you perform GET/POST queries: usually xml or json are used (the latter one being usually preferred since it is more 'compact' and less verbose). This means that you will need to learn how to do the job every time you have to deal with a new vendor, which url object to query, how to retrieve or post the correct configuration payload to obtain the desired result, unless you go for an 'abstraction' tool that unifies syntax (… but does one really exist ?!?!).

REST API

From Wikipedia: “Representational state transfer (REST) is a software architectural style which uses a subset of HTTP. It is commonly used to create interactive applications that use Web services. A Web service that follows these guidelines is called RESTful. Such a Web service must provide its Web resources in a textual representation and allow them to be read and modified with a stateless protocol and a predefined set of operations. This approach allows interoperability between the computer systems on the Internet that provide these services. REST is an alternative to, for example, SOAP as way to access a Web service.”

Practically speaking, we are talking about using protocol ‘http’ and ‘https’ to perform GET (retrieve data) and POST (configure data) queries. The url refers to a specific resource object, something like the following:

https://<remote_device>/api/mo/uni.xml

Xml refers to the requested data format for the posted or retrieved data, the potential alternative is ‘json’. An example of the possible xml body of a post query is the following:

<fvTenant name="ExampleCorp"/>

The same example in ‘json’ format would be the following:

```
{
  "fvTenant": {
     "attributes": {
        "name": "ExampleCorp"
     }
  }
}
```

Such an https post would create a new tenant in ACI fabric. Authenticating into the APIC is done in the same way: performing a POST query, sending the username and password. Even though the syntax regarding the way information are represented is a standard (xml, json, yaml and yang are two others not used for REST APIs), the content is vendor specific. How easy it is to interact with such systems strongly depends on how good is the documentation about it.

A GUI is NOT automation

The above explanation often leads to videos on the web related to orchestration tools, claiming they are doing automation. There are people out there thinking that configuring a bgp session through a CLI after ssh-ing the target devices is NOT automation, while providing the same configuration parameters through a GUI, deploying them and then ssh-ing the device anyway to check that everything is fine, IS automation. Quite funny, at least in my opinion. The advantage of such a tool, would be a multi-vendor tool that depending on the vendor’s target device, adapts the configurations based on the provided parameters, thus saving us from learning both Cisco and Juniper syntax to configure something. An abstraction platform, just like Terraform for Cloud’s infrastructure as a Code. Better than nothing, but not enough since you will IN ANY CASE need to learn the vendor specific commands for troubleshooting

purposes. This is NOT automation at all, it doesn't save time nor money, since the seniority of the people configuring the GUI needs to be the same.

Automation vs Orchestration

What is the difference between the two ?

*'**Automation**, generally speaking, means completing a **single task** or function without human intervention. Executed wisely, automation makes traditionally time-intensive, manual processes more efficient and reliable.'*

*'**Orchestration** is more related to the concept of managing a large-scale virtual environment or network, thus **executing many automated tasks belonging to a workflow**, in the required sequence. Orchestrating the scheduling and integration of automated tasks between complex distributed systems and services—whether on-premise or in the cloud—streamlines and simplifies interconnected workloads, repeatable processes, and operations.'*

No reason to fight about this definition, unless someone starts asking for automation, but is thinking to orchestration as he says that. The final goal is clear, just as the direction. The problem in general, is that configuring a new application nowadays has become too complex, requiring too much time and often many steps to configure all the involved devices. Switches, routers, firewalls, DNS, balancers, vlans, routing, ports opening, in an on-premises environment or partially in a cloud or multi-cloud environment. This process takes months, sometimes even in case we are dealing with just a new vlan. It's probably more a matter of time than just money.

But automating everything is a very difficult and complex task, since it usually requires programming skills, to do configurations that are understood by other people who are networking experts, security experts OR (exclusive) virtualization exerts, Cloud experts. They should all talk, work, cooperate and test things together to achieve the desired result. Despite of the technical figures searched on the web, **there are no experts-of-everything people**. This means that automating EVERY step of the whole process described above, means investing a lot of money, and at least in the short period it will cost you (much) MORE. In the medium term, you would potentially save (much) time, possibly some money (hopefully not firing people, but lowering down your time to market).

If you don't want to hire programmers, you can always buy commercial tools to perform automation/orchestration of all your processes, there's plenty out there claiming to be able to do everything and being multi-vendor, some of them are even free and open source. Ansible, Chef, Puppet, just to mention a few 'automation' tools. Some of them have overlapping areas of usage, for example **Ansible Tower** provides orchestration capabilities, providing the possibility of drawing the job sequence of steps. **Jenkins** is another tool potentially offering similar features, with powerful add-ons and extensions, even offering the scalability of delegating remote devices to perform the required jobs (aka the single tasks to be completed and belonging to the global **workflow**). The more these tools are flexible, powerful and configurable, the more you will need time to customize them to fulfill the required automation goals, to extend them, to write custom scripts or packages that do the 'real automation'.

Automation examples

Some automation examples, coming from my personal working experience:

1. configuring the same command on multiple (hundreds or thousands) devices (simple task)
2. retrieving the output of a sequence of commands from multiple devices (simple task)
3. migrating the access switches (with hundreds of ports each one) of a network provider to newer ones (very complex)
4. migrating the PE devices of a network provider, with hundreds of ports and (sub)interfaces (very complex)
5. migrating a DataCenter from a legacy environment to an SDN based one (very complex)

The difference between the listed examples, is mainly related to the complexity:

· simple tasks that could be flexibly repeated many times in the present and the future (1, 2 above)
· very complex activities, that are potentially performed on many nodes, but are not likely going to be repeated in the future (examples 3, 4 and 5)

Sometimes there is no real choice, especially if you want to survive and maintain a mental brain sanity: automating activity 3 and 4 depends on how many nodes you are going to replace. Even if there are only 20 nodes, it's usually worth it. Automating and writing scripts to create THOUSANDS of objects and configuration inside a DC ACI fabric, to migrate an existing legacy environment, is always worth it. You're probably learning how to do it once, and hopefully re-using it in the future (for other automation stuff based on REST APIs, or other DataCenters to migrate).

Automation saves time, money and service outages

True. But it also depends on the specific use case, and **HOW** automation is done. For sure automation saves TIME, in every possible example. Unless you didn't do something wrong. Unfortunately, every automated activity, even the most simple ones, are based on some data that needs to be provided: in case you need to configure a simple command line on a thousand Cisco devices, if the command line contains a typo, the human error gets replicated by automation on all that thousand devices. In case this has hopefully no service outages consequences, you will need a second job to remove the error and re-configure the correct one. Even in case you have only 6 devices, but providing a service to the whole nation you are serving as a provider, it is always good and safe to split even simple activities in multiple waves, to stay on the safe side, even though they are considered 'safe' because they have been done many times before.

For these reasons, **it is somewhat arguable that automation saves time and leads to less service outages**, even though averagely speaking this remains true. For sure for the complex activities examples we have previously provided (3, 4, 5), writing a script saves a lot of mistakes that are inevitable for repetitive, brain-burning tasks like "take 2000 interfaces on a legacy data center and copy them on an ACI fabric", or copy configurations from an old PE to a new one replacing Gi1/x with Te1/x because the module is different.

The difference can be absolutely be made **by the quality of the automation**: in case of scripts configuring the devices starting for example from an excel file, all the data could be checked for consistency before doing the configurations on REST API enabled devices. The more

complex the environment, the more complex, useful and important will be the pre-configuration checks. This means a lot of work that is being paid back by less hours spent on trivial activities, less time to do them, less errors and service outages. Keep in mind that you can check if an ip address overlaps with another active one when it shouldn't, but it could be impossible to check or say if the ip address has been chosen in the wrong range, if you didn't put the addressing plan logics in it (with the promise of maintaining them during time …). The more checks you want to do, the more it will costs, the more complex will become the custom scripts, packages and modules, potentially arriving to the need of DevOps people managing the software lifecycle. **Automation is not a 'one shot' expense after which you forget about it, doing it in a serious way implies it to become a RECURRENT expense**.

Orchestration … for everyone, everywhere ?

The truth is that probably automation and orchestration are not for everyone, nor for every company. Cloud services can't be provided worldwide without automating everything, since the beginning. Automation IS PART of their core business. Customers use the provided GUI to do what they have to, prices are fixed and the same for everyone, you're not going to ask for a commercial offer, most of the times. If you have the 'power' to do it (i.e. you're big enough) you could potentially do it at the beginning, but you're not asking for discounts every week you add/remove a couple of virtual machines or containers.

If you are a big company, but your core business is another one, things change. Automation could not be a goal of the managers, unless it really leads to saving money. In this case the questions you should ask to yourself are:

- is it worth it ?
- is automation REALLY related to my core business ?
- how many times do I repeat the SAME task every week ?
- how simple are these tasks ?
- are you really sure that all the time taken to 'go live' is due to the slowness in configuring the required network and security stuff, and not to all your internal processes ? in which proportion the delay is due to one or the other ?

Real life examples

If you are a service provider and every week you have to configure 50 dwdm optical circuits (new ones, or deleting old ones), you will absolutely gain something in automating the configurations. You could create a simple template excel file, with the few required information about the circuits (i.e. ingress device/port, egress device/port, bandwidth, …). People writing these information could be not senior people, they could even not be aware at all of the devices and the technology that is used for dwdm configurations. The automation script will ideally check for data inconsistencies (for example wrong or not existent ports) and will perform all the required configurations, returning a final result. This is (except for the consistency checks) an example published on github by Cisco, and doing exactly what has been described:

https://github.com/CiscoSE/Automating-Service-Provisioning-with-EPNM-API

This will be a cost in the short term period, but it's worth it. But what if you do a couple of circuits every month ? Or even every week ? Is it still worth it ? Probably not.

One other example could be the configurations inside an ACI DataCenter: you could potentially configure anything, vlans, physical interfaces, dynamic routing, contracts. But what can REALLY be automated ? does it make sense automating the creation of bgp sessions ? probably not. Automating contracts ? maybe, even though creating an excel file containing the necessary data could be complex, or even 'dangerous' (too difficult) for the people who need to fill-in the data. Could you automate the configuration of physical interfaces ? probably yes. Could you automate the configurations of Bridge Domains, EPG, the physical interfaces to which they are associated and the related vlan encapsulations ? probably yes. There are always choices and decisions to take. Once again the question is … how many times a week do you perform this kind of activities ? how many ports or new vlans do you configure ? there's a clear difference between a cloud provider and the on-premise DataCenter of a private company.

Hereafter follows a Python script that takes data from an excel file and configures ACI. Consistency checks are not made on provided data, so you would need to rely on ACI checks, or develop your custom ones.

https://github.com/carlmontanari/acipdt