DevNet Experts.

Topic- Jinja2

+91 9892028199

devnetexperts@gmail.com

DevNet Experts

OVERVIEW

- Jinja2 Introduction
- Variable Substitution
- Ansible with Jinja2
- Jinja2 Templating
- Use cases

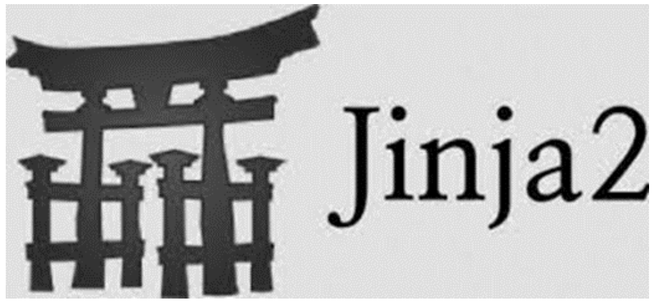
Introduction.

Jinja2 is a feature rich templating language widely used in the Python ecosystem.

- Can be used directly in Python programs
- Can also be used in a wide range of applications as their template rendering engine e.g.

1. Web frameworks like Django, Flask etc.
2. Configuration management tools like Ansible, Saltstack
3. Static site generator tools like Pelican and so on.

- Let’s try to put things in perspective.



Template

- Imagine Cisco having to print thousands of such certificates for the vast number of certifications they offer.
- They create a template with placeholders for the name of the person, certification, logo, dates etc.
- The things that tend to change from one certificate to the other.
- The fixed part, along with placeholders becomes a template.

Template with Variable names

Placeholders in computer terms are nothing but variables.

Back to Agenda Page

Data Stored in Variables.

DevNet Experts

Name

Ehsan Momeni Bashusqeh

Certification name

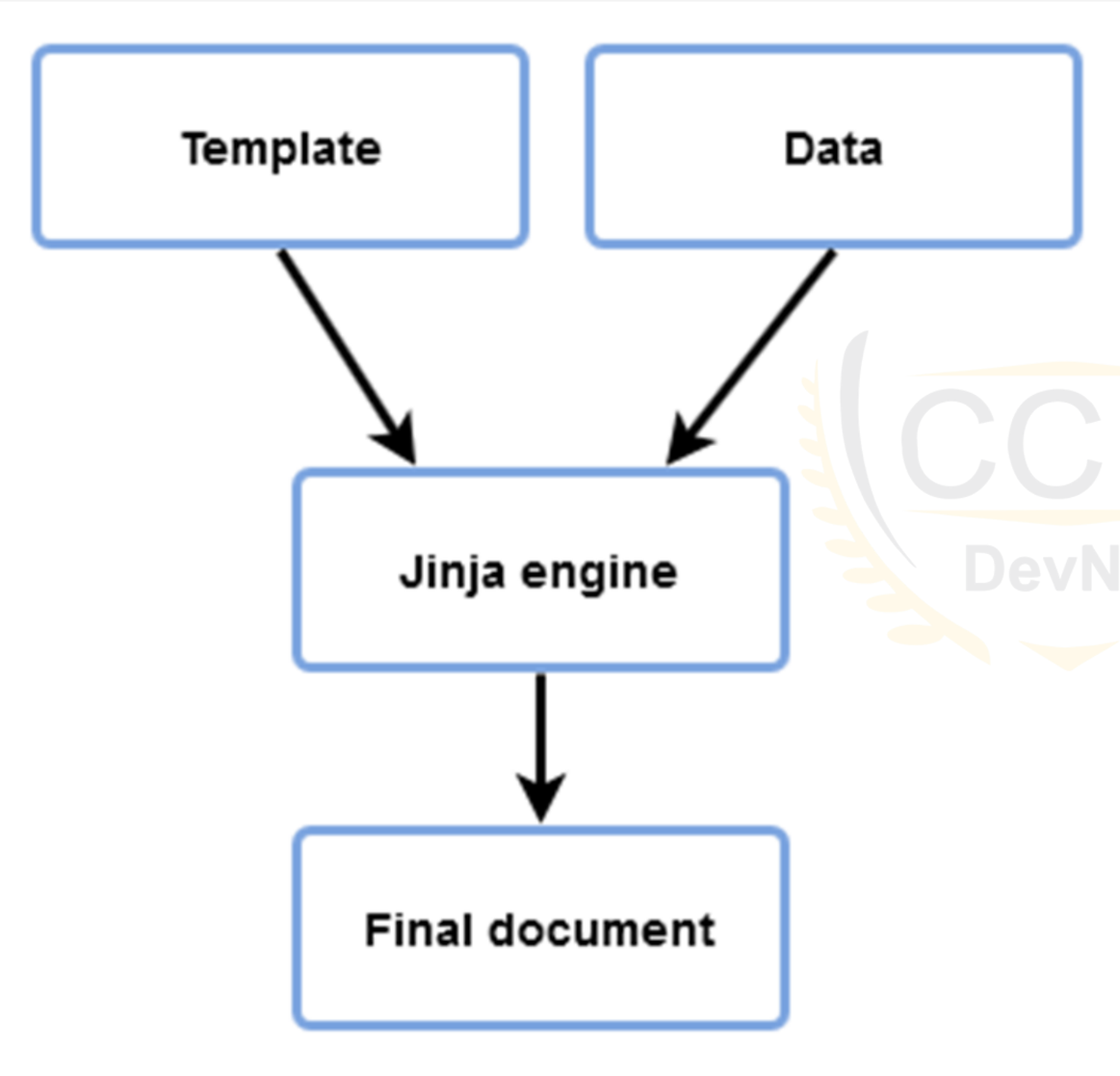
Cisco Certified Network Associate Routing and Switching

Logo

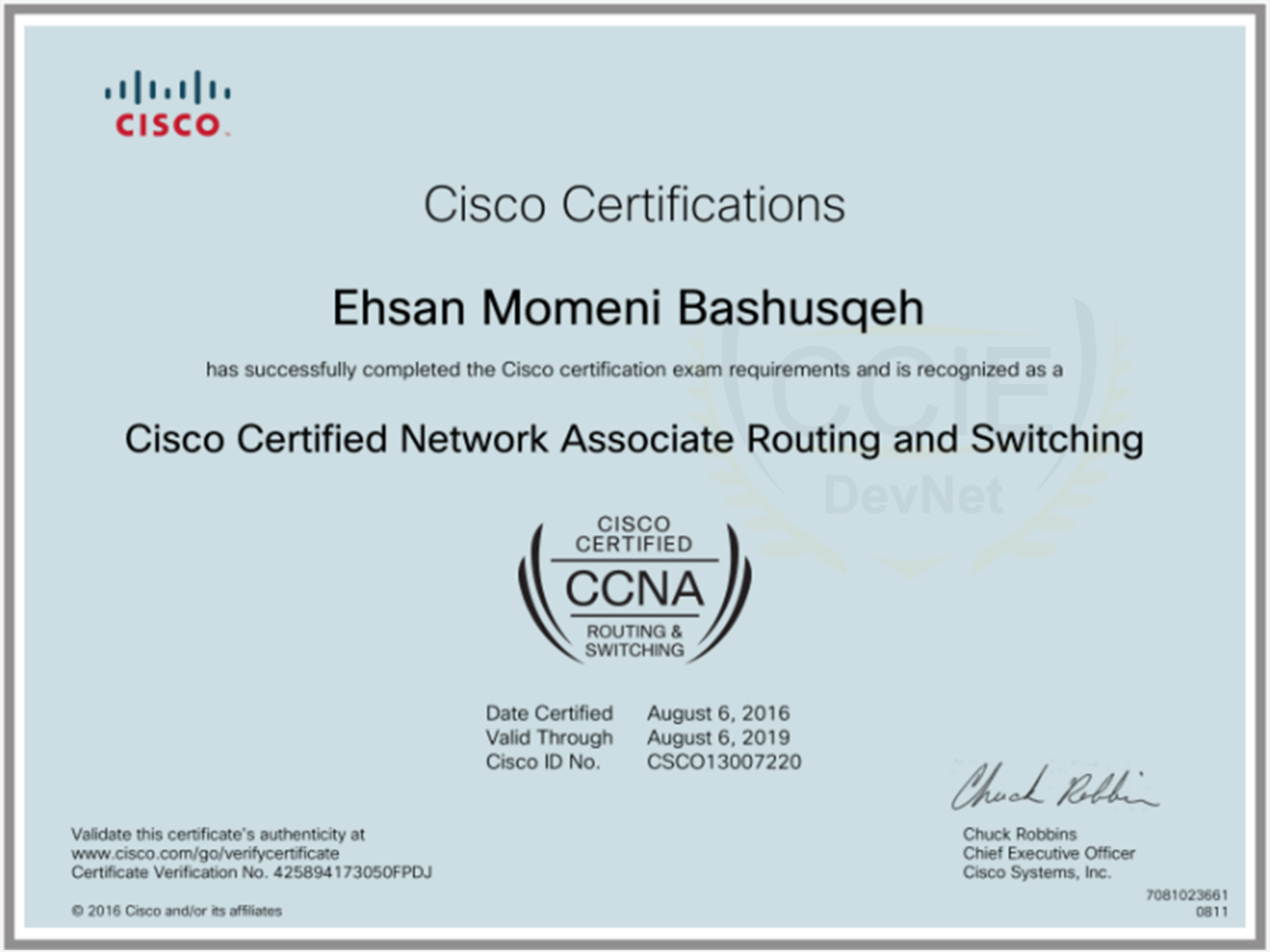
cert_date
valid_date
cisco_id

August 6, 2016
August 6, 2019
CSCO13007220

Enters Jinja2



Final Document.



Cisco Certifications

Ehsan Momeni Bashusqeh

has successfully completed the Cisco certification exam requirements and is recognized as a

Cisco Certified Network Associate Routing and Switching

CISCO CERTIFIED CCNA ROUTING & SWITCHING

Date Certified August 6, 2016
Valid Through August 6, 2019
Cisco ID No. CSCO13007220

Validate this certificate's authenticity at
www.cisco.com/go/verifycertificate
Certificate Verification No. 425894173050FPDJ

Chuck Robbins
Chief Executive Officer
Cisco Systems, Inc.

7081023661
0811

Ansible with Jinja2

- As we know Ansible has a large collection of modules, Jinja2 module also comes preinstalled with Ansible.
- Ansible provides variables to the templates and renders them using the template module which in turn calls the rendering engine of Jinja2.
- Template rendering happens on Ansible controller
- Rendered task is then sent to the target machine for execution.
- This is done to minimize the package requirement on target machine.
- This also limits the amount of data Ansible passes to the target machine.

REST APIs (Cont.)

- Follow Object Oriented programming paradigm of noun-verb.
- There is an object or entity (noun) and we perform some actions on that (verb).
- Verbs are the actions performed on the nouns or resources.
- Actions are performed by sending a request to the server.
- And results of the actions are sent back to the client as response.
- This request and response happens via one of the machine-readable data interchange formats we discussed earlier like XML, YAML, JSON etc.

Jinja2 Templating.

- Jinja2 needs the following source ingredients to work

1. Template
2. Data

- Data can come from various sources like

1. JSON data returned by API
2. Loaded from static YAML file
3. Python dictionary defined in our application

- Basic idea is to identify static and dynamic parts of the documents
- Dynamic parts are parametrized, so they change according to the data passed
- Hence multiple versions of the document are created with static part being the same and dynamic part changing as per the data passed

A more relevant use case - BGP Configuration.

```
router bgp 45000
 router-id 172.17.1.99
 bgp log-neighbor-changes
 neighbor 192.168.1.2 remote-as 40000
 neighbor 192.168.3.2 remote-as 50000
 address-family ipv4 unicast
  neighbor 192.168.1.2 activate
  network 172.17.1.0 mask 255.255.255.0
  exit-address-family
```

Sample target config we want to generate.

- Shown above is a short snippet of CIsco IOS configuration
- First we identify which part of the above snippet is static and which parts change between devices
- Typically ASNs, IP Addresses, address family type etc. change between the devices
- The parts that change are converted into variables to be substituted with actual data when template is rendered at runtime

A more relevant use case - BGP Configuration


```
router bgp {{ local_asn }}
 router-id {{ router_id }}
 bgp log-neighbor-changes
 neighbor {{ neighbor_id_1 }} remote-as {{ remote_asn_1 }}
 neighbor {{ neighbor_id_2 }} remote-as {{ remote_asn_2 }}
 address-family ipv4 unicast
neighbor {{ neighbor_id_1 }} activate
network {{ network }} mask {{ net_mask }}
exit-address-family
```

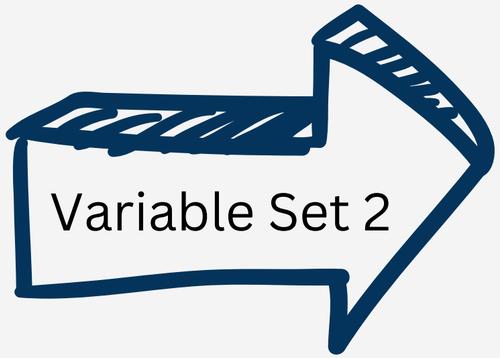
Actual values replaced with variables. This becomes a template now.

- In Jinja2 anything found between a pair of double opening and closing curly braces (“{{”, “}}”), known as delimiters, will be evaluated and replaced by the engine
- The templating engine expects to find a variable with the same name in the list of variables
- The variable name in the template will then be replaced with the value from the data file which can be a JSON file, YAML file, Python dictionary etc.

A more relevant use case - BGP Configuration

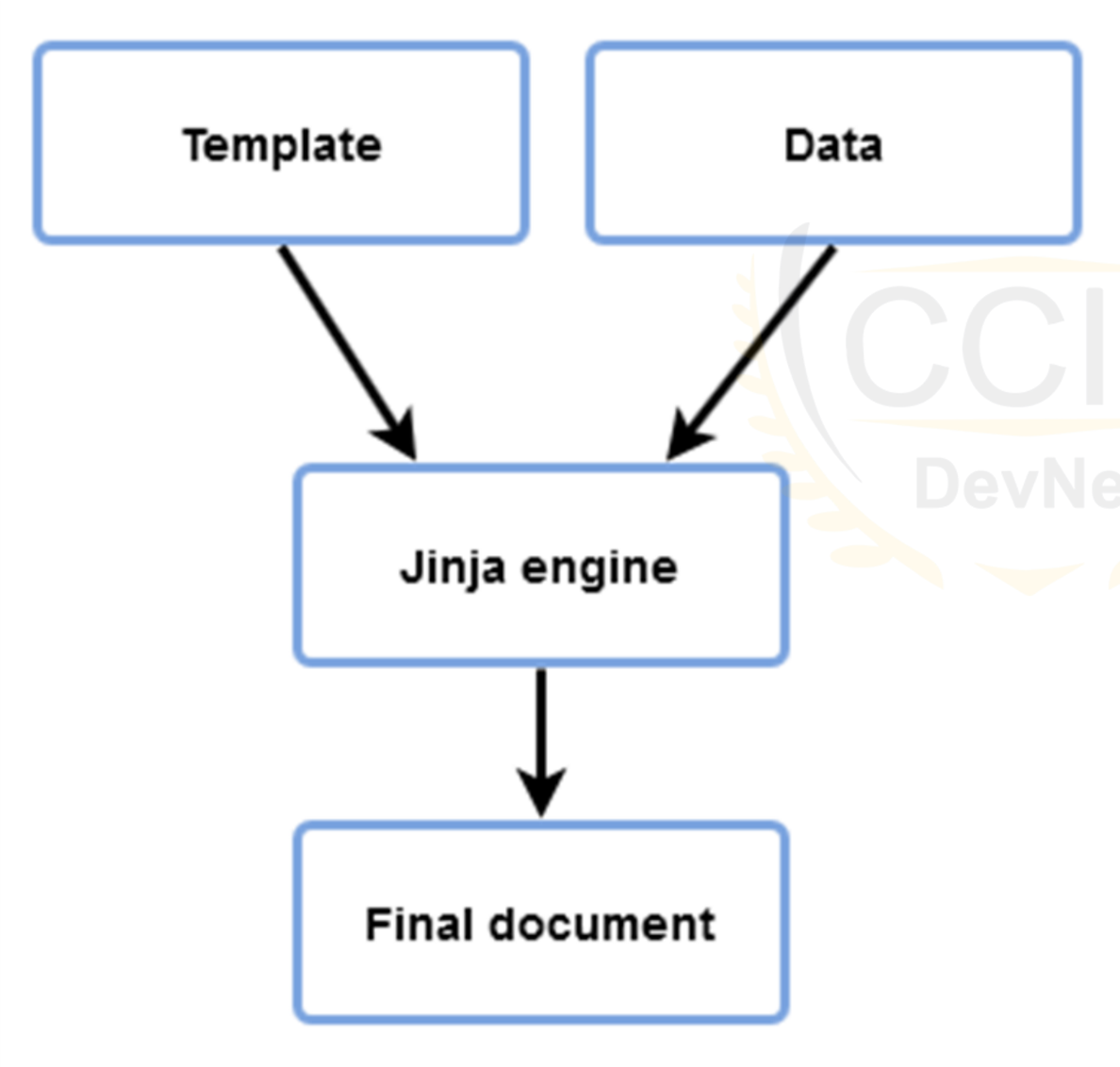
local_asn: 45000
router_id: 172.17.1.99
neighbor_id_1: 192.168.1.2
neighbor_id_2: 192.168.3.2
remote_asn_1: 40000
remote_asn_2: 50000
network: 172.17.1.0
net_mask: 255.255.255.0

- When we substitute Variable Set 1 into the template, we get first set of BGP config commands
- Similarly when we substitute Variable Set 2 into the template, we get second set of BGP config commands



local_asn: 95000
router_id: 172.17.1.200
neighbor_id_1: 192.168.10.200
neighbor_id_2: 192.168.20.200
remote_asn_1: 70000
remote_asn_2: 80000
network: 172.17.1.0
net_mask: 255.255.255.0

Enters Jinja2



DEMO.