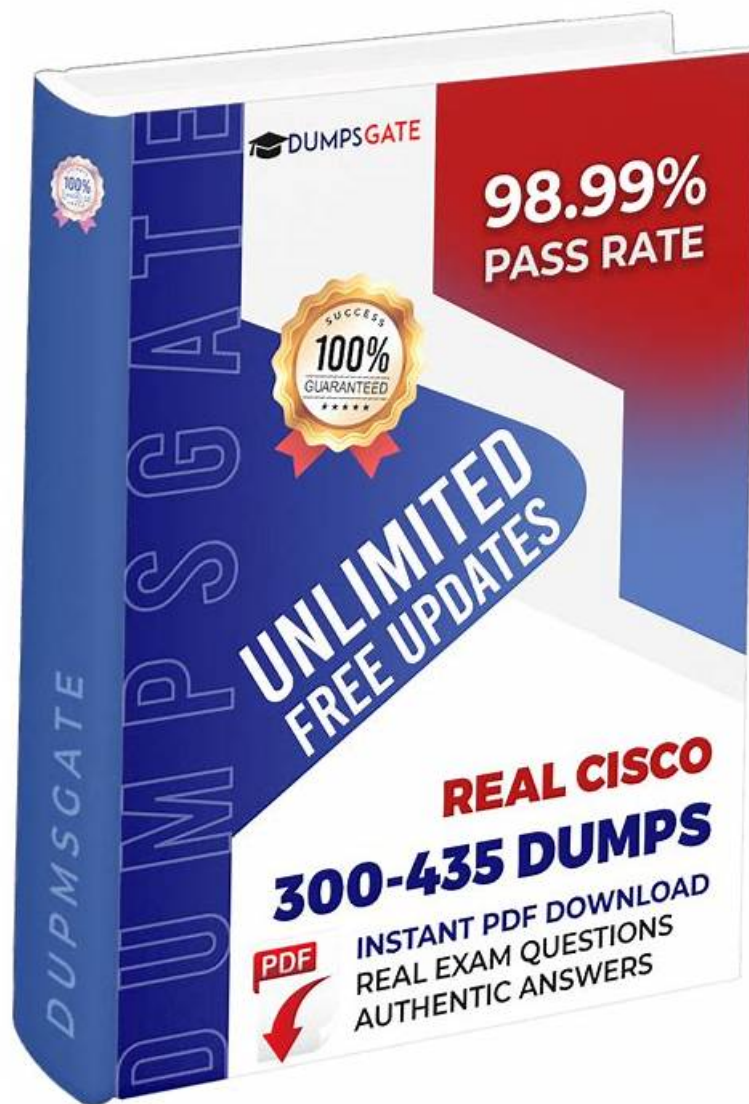


Use Latest Cisco 300-435 Dumps And Gain Brilliant Scores



P.S. Free & New 300-435 dumps are available on Google Drive shared by Dumpcollection: https://drive.google.com/open?id=1bE48eV9QN2zXoIQXpo4VzsmNU_kQeD3F

The Automating and Programming Cisco Enterprise Solutions (300-435) practice test software also keeps a record of attempts, keeping users informed about their progress and allowing them to improve themselves. This feature makes it easy for 300-435 desktop-based practice exam software users to focus on their mistakes and overcome them before the original attempt. Overall, the Windows-based Automating and Programming Cisco Enterprise Solutions (300-435) practice test software has a user-friendly interface that facilitates candidates to prepare for the Automating and Programming Cisco Enterprise Solutions (300-435) exam without facing technical issues.

Cisco 300-435 exam tests the candidates' knowledge and skills in implementing automation workflows, programming concepts, and Python programming. 300-435 exam covers a wide range of topics, including network programmability foundations, APIs, network device programmability, automation tools, and Cisco SD-WAN. 300-435 Exam is designed to validate the candidate's ability to automate and program Cisco Enterprise Solutions using Cisco platforms and APIs.

>> 300-435 Exam Lab Questions <<

Exam 300-435 Cram Questions & 300-435 Exam Topic

Dumpcollection trained experts have made sure to help the potential applicants of Automating and Programming Cisco Enterprise Solutions certification to pass their Automating and Programming Cisco Enterprise Solutions exam on the first try. Our PDF format carries real Cisco 300-435 Exam Dumps. You can use this format of Cisco 300-435 actual questions on your smart devices.

Cisco Automating and Programming Cisco Enterprise Solutions Sample Questions (Q56-Q61):

NEW QUESTION # 56

Information about a rebooted device needs to be displayed with an ID of 260faff9-2d31-4312-cf96-143b46db0211 using the Cisco SD-WAN vManage Administration APIs. The API documentation states that deviceId is a required request parameter. Fill in the blank to create the REST call.

`https://vmanage-ip-address:8443/dataservice/device/action/reboot` `260faff9-2d31-4312-cf96-143b46db0211`

Answer:

Explanation:
"deviceId":

NEW QUESTION # 57

Drag and drop the commands to the Ansible playbook that applies configuration to an interface on a Cisco IOS XE device. Not all options are used.

Answer Area

ioscmd

parents

iosconfig

interface

iosxe

ios_config

```
- name: configure interface settings
  :
    lines:
      -ip address 172.31.1.1 255.255.255.0
      -no shutdown
      : interface GigabitEthernet1/0
```

Answer:

Explanation:

Answer Area

ioscmd

parents

iosconfig

interface

iosxe

ios_config

```
- name: configure interface settings
  ios_config:
    lines:
      -ip address 172.31.1.1 255.255.255.0
      -no shutdown
      parents: interface GigabitEthernet1/0
```

```
- name: configure interface settings
  ios_config:
    lines:
      -ip address 172.31.1.1 255.255.255.0
      -no shutdown
  parents: interface GigabitEthernet1/0
```

NEW QUESTION # 58

Which Python snippet receives a Meraki webhook request?

```
@app.route('/mynet/webhook', methods=['PUT'])
@app.accept_body(WebhookSchema)
def receive_webhook(**kwargs):
    send_sms_alert(kwargs['alertType'])

@app.route('/mynet/webhook', methods=['GET'])
@app.accept_body(WebhookSchema)
def receive_webhook(**kwargs):
    send_sms_alert(kwargs['alertType'])

@app.route('/mynet/webhook', methods=['PATCH'])
@app.accept_body(WebhookSchema)
def receive_webhook(**kwargs):
    send_sms_alert(kwargs['alertType'])

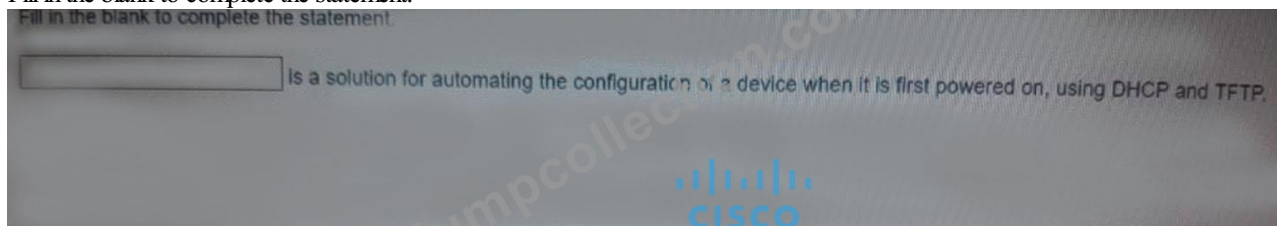
@app.route('/mynet/webhook', methods=['POST'])
@app.accept_body(WebhookSchema)
def receive_webhook(**kwargs):
    send_sms_alert(kwargs['alertType'])
```

- A. Option D
- B. Option B
- C. Option C
- D. Option A

Answer: A

NEW QUESTION # 59

Fill in the blank to complete the statement.



Answer:

Explanation:
ZTP

NEW QUESTION # 60

Drag and Drop Question

Refer to the exhibit. Drag and drop the code from the bottom onto the box where the code is missing to complete the XML instance of this YANG module. Options may be used more than once. Not all options are used.

```

module interfaces {
  typedef dotted-quad {
    type string {
      pattern
        '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])\.([0-9]|1[0-9]|2[0-4]|25[0-5])\.'{3}'
        + '([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-4][0-9]|25[0-5])';
    }
    description
      "Four octets written as decimal numbers and
        separated with the '.' (full stop) character.";
  }
  container interfaces {
    list interface {
      key "name";
      leaf name {
        type string;
        mandatory "true";
        description
          "Interface name.";
      }
      leaf address {
        type dotted-quad;
        mandatory "true";
        description
          "Interface IP address.";
      }
      leaf subnet-mask {
        type dotted-quad;
        mandatory "true";
        description
          "Interface subnet mask.";
      }
      leaf enabled {
        type boolean;
        default "false";
        description
          "Enable or disable the interface.";
      }
    }
  }
}

```

```

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  < [ ] xmlns="http://example.com/interfaces">
    < [ ] >
      < [ ] >GigabitEthernet 0/0/0</ [ ] >
      <address>10.10.10.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </ [ ] >
  </ [ ] >
</data>

```

list	name	id	leaf name
interface	description	interfaces	

Answer:

Explanation:

```

<data xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  < interfaces xmlns="http://example.com/interfaces">
    < interface >
      < name >GigabitEthernet 0/0/0</ name >
      <address>10.10.10.1</address>
      <subnet-mask>255.255.255.0</subnet-mask>
    </ interface >
  </ interfaces >
</data>

```

list	name	id	leaf name
interface	description	interfaces	

Explanation:

interfaces: Matches the YANG container interfaces.

interface: Matches the YANG list interface.

name: Matches the YANG leaf name defined as the list key.

The correct closing tags follow the same logic and structure:

- </interface> closes the list item.
- </interfaces> closes the container.

• • • • •

Exam 300-435 Cram Questions: https://www.dumpcollection.com/300-435_braindumps.html

- What's more, part of that Dumpcollection 300-435 dumps now are free: https://drive.google.com/open?id=1bE48eV9QN2zXoIQXpo4VzsmNU_kQeD3F