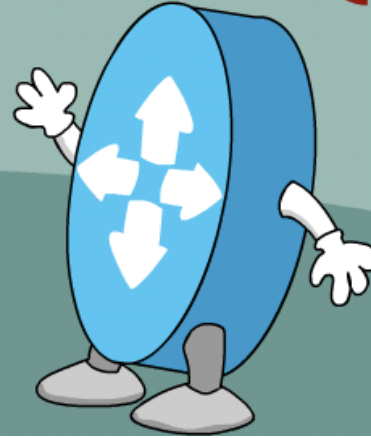




# NETDEVOPS {LIVE!}



DEVNET

# Laptop Tips and Tricks for the NetDevOps Engineer

Hank Preston, ccie 38336 R/S  
NetDevOps Engineer, DevNet Sandbox  
Twitter: @hfpreston

Season 2, Talk 5

<https://developer.cisco.com/netdevops/live>

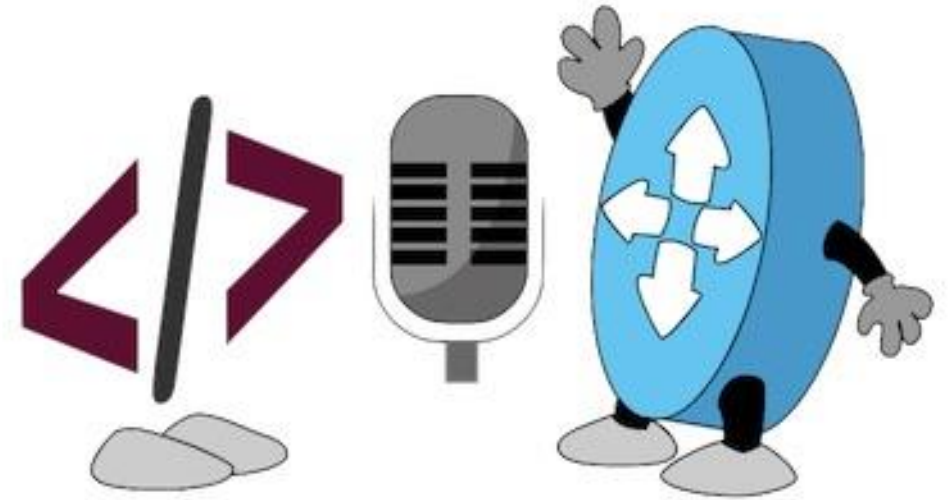


<http://cs.co/ndl>

Help us track NetDevOps Live Interest!

# What are we going to talk about?

- Don't be bashful, Love thy terminal
- Shhh... SSH like a pro
- Oscar worthy bash scripts!  
(and other bash-y stuff)
- Make your Apps Dance



# Before we start...

- ✓ What follows are all personal opinions, some tips might not be for you
- ✓ Some tips are very basic, and you might already know it... For others they'll be new
- ✓ I learn new stuff all the time... there are likely great tips I don't know, share them!

Don't be bashful, Love thy  
terminal

```

osxe1# guestshell run bash
admin@guestshell ~]$
admin@guestshell ~]$ bash --version
GNU bash, version 4.2.46(1)-release (x86_64-redhat-linux-gnu)
Copyright (C) 2011 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
[admin@guestshell ~]$ ls -l
total 0
[admin@gu

```

```

MINGW64/c/Users/hapresto
hapresto@HAPRESTO-TCCVH MINGW64 ~
$ bash --version
GNU bash, version 4.4.12(1)-release (x86_64-pc-msys)
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
hapresto@HAPRESTO-TCCVH MINGW64 ~
$

```

*Bash isn't the only terminal... but it is everywhere and you need to be comfortable with it*

```

hapresto@HAPRESTO-TCCVH:~$ ip link
link/ether 54:e1:ad:66:41:b2
link/ether f8:59:71:c9:4a:95
eth3: <BROADCAST,MULTICAST,UP> mtu 1500 group default qlen 1

```

```

sbx-n9kv-ao# run bash
bash-4.3$ ip route
127.1.0.0/16 dev veobc proto kernel scope link src 127.1.1.1
127.1.2.0/24 dev veobc proto kernel scope link src 127.1.2.1
172.16.1.0/30 dev Eth1-5 proto kernel scope link src 172.16.1.1
172.16.100.0/24 dev Vlan100 proto kernel scope link src 172.16.100.1
172.16.101.0/24 dev Vlan101 proto kernel scope link src 172.16.101.1
172.16.102.0/24 dev Vlan102 proto kernel scope link src 172.16.102.1

```

# May the “source” be with you

- Ever wonder what “source” does in bash?
- Given a bash script, what’s the difference between running and “sourcing”

## Answer

- Running a script generates a new shell instance
- Sourcing runs the script in your current shell
- So what?
  - Environment variables, directory changes

File: env\_demo1.sh

```
#!/bin/bash

echo "This is a sample bash script."
echo "I am going to set up an env var called 'DEMO_VAR1'"

export DEMO_VAR1="New ENV VAR!"

echo "Okay, see if it worked with 'echo \${DEMO_VAR1}'"
echo ""
```

```
# Option 1
$ chmod +x env_demo1.sh
$ ./env_demo1.sh
$ echo ${DEMO_VAR1}
```

```
# Option 2
$ source env_demo1.sh
$ echo ${DEMO_VAR1}
```

# Which worked?

[dev.tips/env\\_demo1.sh](https://dev.tips/env_demo1.sh)

# Aliases – for more than restoring “wr mem” to NX-OS

- Create shortcuts for common, long, or complicated commands
- Git commands
- File management
- VPN Connections
- Anything

```
sbx-n9kv-ao# wr me
```

```
^
```

```
% Invalid command at '^' marker.
```

```
sbx-n9kv-ao# conf t
```

```
sbx-n9kv-ao(config)# cli alias name wr copy running-config startup-config
```

```
sbx-n9kv-ao(config)# end
```

```
sbx-n9kv-ao# wr
```

```
[#####] 100%
```

```
Copy complete, now saving to disk (please wait)...
```

```
Copy complete.
```



# Bash Aliases

```
# A few handy bash aliases

# Display file list with colors, sizes, and in list format
alias ll='ls -FGlAhp'

# Handy aliases to find resource hogs
alias memHogsTop='top -l 1 -o rsize | head -20'
alias memHogsPs='ps wwaxm -o pid,stat,vsize,rss,time,command | head -10'
alias cpu_hogs='ps wwaxr -o pid,stat,%cpu,time,command | head -10'

# Some networking aliases for macOS
alias flushDNS='dscacheutil -flushcache'
alias openPorts='sudo lsof -i | grep LISTEN'

# Remove all exited/stopped docker containers
alias docker_rm='docker rm -v $(docker ps -aq -f status=exited)'
```

[dev\\_tips/alias\\_examples.sh](https://github.com/devnet/devnet/blob/master/dev_tips/alias_examples.sh)

# The bash prompt

- Environment Variable PS1
- Series of codes and functions generate what you see
- You can make it whatever you want
- `export PS1=\W\ $`
  - Working directory and dollar sign
- Adding some git info
  - <https://github.com/magicmonty/bash-git-prompt>

The screenshot shows the EZPROMPT Easy Bash PS1 Generator interface. It is divided into four main steps:

- 1.) Pick the elements you want to use in your prompt.** This section has four tabs: "Basic Elements", "Status Elements", "Date & Time Elements", and "Extra Characters". Under "Basic Elements", several elements are selected: Username, Hostname, FQDN, Shell, Shell Version, Shell Release, Path To Current Directory, Current Directory, and a dollar sign (\$).
- 2.) Select colors and rearrange elements here.** This section shows a preview of the prompt elements: Username (with a dropdown arrow), Hostname, Git Status, and \$ (with a green background). Below this are controls for "Delete", "Reset", "FG" (with a color dropdown), "BG" (with a color dropdown), "Reset All", and "Delete All".
- 3.) Preview the output.** This section shows a black terminal window with the prompt: `user host[master]$`.
- 4.) Copy and paste the code into your bashrc.** This section contains a code block with the following content:

```
# get current branch in git repo
function parse_git_branch() {
  BRANCH=$(git branch 2> /dev/null | sed -e '/^[*]/d' -e 's/* \(.*\)/1/')
  if [ ! "${BRANCH}" == "" ]
  then
    STAT=$(parse_git_dirty)
    echo "${BRANCH}${STAT}"
  else
    echo ""
  fi
}

# get current status of git repo
function parse_git_dirty {
```

<http://ezprompt.net>

# .bash\_profile (and .bashrc on Linux)

- Auto-load and customize your bash environment
- Terminal settings, prompts, colors, aliases, just about anything
- Many many examples of customizations online

```
199 # -----
200 # 6. NETWORKING
201 # -----
202
203 alias myip='curl ip.appspot.com'           # myip:      Public facing IP Address
204 alias netCons='lsof -i'                   # netCons:   Show all open TCP/IP sockets
205 alias flushDNS='dscacheutil -flushcache'  # flushDNS:  Flush out the DNS Cache
206 alias lsock='sudo /usr/sbin/lsof -i -P'    # lsock:     Display open sockets
207 alias lsockU='sudo /usr/sbin/lsof -nP | grep UDP' # lsockU:    Display only open UDP sockets
208 alias lsockT='sudo /usr/sbin/lsof -nP | grep TCP' # lsockT:    Display only open TCP sockets
209 alias ipInfo0='ipconfig getpacket en0'     # ipInfo0:   Get info on connections for en0
210 alias ipInfo1='ipconfig getpacket en1'     # ipInfo1:   Get info on connections for en1
211 alias openPorts='sudo lsof -i | grep LISTEN' # openPorts: All listening connections
212 alias showBlocked='sudo ipfw list'         # showBlocked: All ipfw rules inc/ blocked IPs
213
214 # ii: display useful host related informaton
215 # -----
216 ii() {
217     echo -e "\nYou are logged on ${RED}$HOST"
218     echo -e "\nAdditional information:$NC " ; uname -a
219     echo -e "\n${RED}Users logged on:$NC " ; w -h
220     echo -e "\n${RED}Current date :$NC " ; date
221     echo -e "\n${RED}Machine stats :$NC " ; uptime
222     echo -e "\n${RED}Current network location :$NC " ; sselect
223     echo -e "\n${RED}Public facing IP Address :$NC " ; myip
224     #echo -e "\n${RED}DNS Configuration:$NC " ; scutil --dns
225     echo
226 }
227
```

[https://natelandau.com/my-mac-osx-bash\\_profile/](https://natelandau.com/my-mac-osx-bash_profile/)

Shhh... SSH like a Pro

# Authentication Power User – Use the Cert Hank!

- Passwords are so 1980s...
- Let's see how to use certificate based authentication

```
[developer@devbox ~]$ssh-keygen

Generating public/private rsa key pair.
Your identification has been saved in
/home/developer/.ssh/id_rsa.
Your public key has been saved in
/home/developer/.ssh/id_rsa.pub.

[developer@devbox ~]$ssh-copy-id
developer@10.10.20.20
Number of key(s) added: 1

Now try logging into the machine, with:
    "ssh 'developer@10.10.20.20'"
```

# SSH Certificate Notes

- Create a key pair
  - `ssh-keygen`
  - Default location for keys: `~/.ssh`
- Copy key to another server
  - `ssh-copy-id`
  - Adds to `~/.ssh/authorized_keys`
- Linux SSH Server Config File
  - `/etc/ssh/sshd_config`
- IOS SSH Auth
  - `ip ssh pubkey-chain`
    - `username admin`
    - `key-string`
    - `STRING`
    - `STRING`
  - Bash command to wrap key to length
    - `fold -w 48 .ssh/id_rsa.pub`
- NX-OS SSH Auth
  - `username admin sshkey KEY`

# ~/.ssh/config - SSH Host Aliases and Properties

- SSH “aliases” for Hosts
  - Make it easier to connect to common devices
- Enable non-standard ciphers and algorithms
- Wild Cards for settings

```
Host sbx-iosxe
  HostName ios-xe-mgmt.cisco.com
  User root
  Port 8181
  IdentityFile ~/.ssh/id_rsa
Host sbx-nxos
  HostName sbx-nxos-mgmt.cisco.com
  User admin
  Port 8181
Host OldSwitch
  HostName 10.10.2.1
  Ciphers aes128-cbc
  KexAlgorithms +diffie-hellman-group1-sha1
```

# ~/.ssh/config - Those pesky changing lab certs...

- How to handle hosts where you fully expect keys to change
- You can use wildcards in names

```
Host 172.20.*.*  
  StrictHostKeyChecking no  
  UserKnownHostsFile=/dev/null
```



Oscar worthy bash scripts!  
(and other bash-y stuff)

# Curl – REST APIs and Web Calls without Postman

- Check if service is up
  - Status Code
- Get some data to process
- Get token or ID for another request

```
$ curl -sk \  
-u 'root:D_Vay!_10&' \  
-H "Accept: application/yang-data+json" \  
"https://10.10.20.48/restconf/data/ietf-interfaces:interfaces/interface=GigabitEthernet1" \  
| python -c 'import sys, json;  
print(  
    json.load(sys.stdin) ["ietf-interfaces:interface"] ["ietf-ip:ipv4"] ["address"] [0] ["ip"]  
    )'  
10.10.20.48
```

[dev\\_tips/curl\\_examples.sh](https://dev_tips/curl_examples.sh)

# Bash Script Example 1

- Using curl in bash script
- Creating variable based on output
- Conditional processing based on value

```
status_code=$(curl -ks \  
-w "%{http_code}" \  
-o /dev/null \  
-u 'root:D_Vay!_10&' \  
-H "Accept: application/yang-data+json" \  
"https://10.10.10.48/restconf/data/ietf-  
interfaces:interfaces"  
)  
  
if [ $status_code -eq 200 ]  
then  
    echo "✅ Yep, it's up and returning a 200. "  
else  
    echo "❌ Nope, it returned a $status_code"  
fi
```

[devtips/bash\\_script\\_ex1.sh](#)

# Bash Script Example 2

- Creating a bash function
- Using while loop to wait until ready

```
# Function to get the status code
get_status_code() {
code=$(curl -ks \
-w "%{http_code}" \
-o /dev/null \
-u 'root:D_Vay!_10&' \
-H "Accept: application/yang-data+json" \
"https://10.10.10.10/restconf/data/ietf-
interfaces:interfaces"
)
echo $code
}

# Loop until status code is 200, or 10 tries
while [ $status_code -ne 200 -a $COUNTER -lt 10 ]; do
sleep 30
status_code=$(get_status_code)
COUNTER=$((COUNTER+1))
done
```

[devtips/bash\\_script\\_ex2.sh](#)

# Bash Script Example 3

- Asking user for input
- Checking exit code with special \$?

```
echo "What interface would you like to check?"
echo "For example 'GigabitEthernet1'"
read interface_name

ip=$(curl -sk \
-u 'root:D_Vay!_10&' \
-H "Accept: application/yang-data+json" \
"https://10.10.10.10/restconf/data/ietf-
interfaces:interfaces/interface=${interface_name}" \
| python -c 'import sys, json;
print(
    json.load(sys.stdin)["ietf-
interfaces:interface"]["ietf-
ip:ipv4"]["address"][0]["ip"]
)')

# Make sure the last command worked
if [ $? -ne 0 ]
then
    echo "Something went wrong, exiting."
    exit 1
fi
```

[devtips/bash\\_script\\_ex3.sh](#)

# Bonus Bash Example

- Processing all files in a directory
- Here `ncs_load` is ran for EVERY file in the `device_configs/` directory

```
#!/bin/bash

for f in device_configs/*
do
    echo "Processing file $f"
    ncs_load -u admin -m -l $f
done
```

# Make things easier with Makefiles

- Most often used for code compilation workflows
- But can make any workflow easier
- Simplify any repeated task for yourself, or others working on your project
- make dev, make clean, make whatever

[devtips/Makefile](https://devtips.cisco.com/Makefile)

```
dev: virl nso virl-generate nso-sync-from
```

```
clean: nso-clean virl-clean genie-clean
```

```
virl:
```

```
@echo "Starting the VIRL topology."  
-@virl up --provision virlfiles/2-ios-router
```

```
virl-clean:
```

```
@echo "Tearing down VIRL topology."  
-@virl down  
-@rm -Rf default_testbed.yaml
```

```
nso:
```

```
@echo "Setting up NSO dev instance."  
@ncs-setup --package cisco-ios --dest .  
@ncs
```

```
nso-clean:
```

```
@echo "Destroying NSO dev instance."  
-@ncs --stop
```

# Makefile Notes

- - before command to ignore error
- @ before command to NOT print command to screen
- Creates it's own shell, set any environment variables within Makefile
- Organize and group commands together into common workflows



Make your Apps Dance

# Application Package Managers

- Installing applications is a necessary evil, make it as easy as possible
- Repeatability and script-ability should be a goal
- The less mouse and click, the better
- RedHat, CentOS, Other “EL” Linux
  - yum
- Ubuntu and other Debian
  - apt
- macOS
  - Homebrew - <http://brew.sh>
- Windows
  - Chocolatey - <https://chocolatey.org>

# Extend your IDEs with Plug-Ins... to a point

- Language syntax checking and display coloring
- Handy tools for diffs, data
- Integrations to cloud services
- So much more
- *Potential Risks*
  - *Performance*
  - *Security Issues (Do you trust the community)*
- VS Code
  - <https://marketplace.visualstudio.com>
- Atom
  - <https://atom.io/packages>
- Sublime
  - <https://packagecontrol.io>

# Peak under the hood with Developer Mode

- Web applications are everywhere today
- Displaying a nice GUI requires API calls, JavaScript, Cookies, and more
- You can get a lot of info about how stuff works by seeing what happens when you interact with a page
- Developer tools on all major browsers today

The screenshot shows the Cisco DNA Center web application interface. The main content area displays a table of devices with columns for Device Name, IP Address, Reachability Status, and Uptime. The table lists several devices, including 'Adam\_TEST', 'asr1001-x', 'cat\_9k\_1.marius.x-trem.ro', and 'N/A'. The 'cat\_9k\_1.marius.x-trem.ro' device is highlighted, and a blue callout box labeled 'Make a Wish' points to its entry.

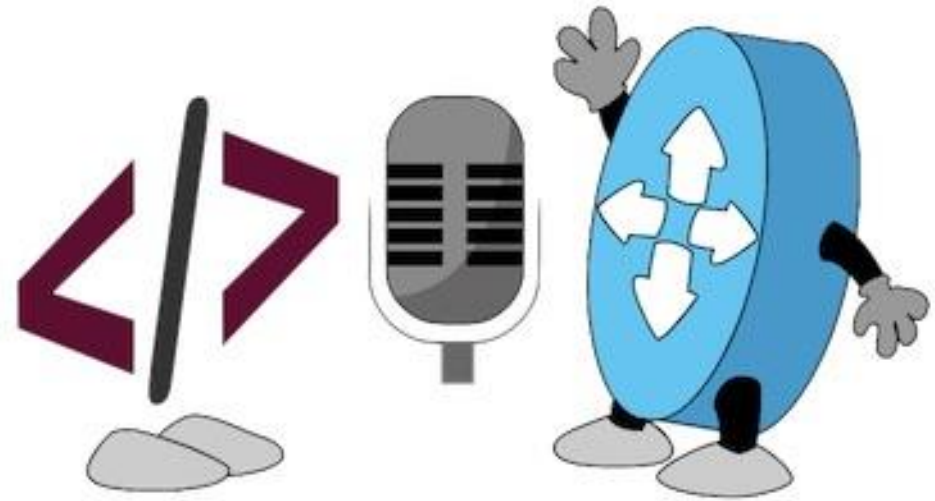
The browser's developer tools are open, showing the Network tab. The selected request is an XHR call to the API endpoint `https://sandboxnac.cisco.com/inventory/devices`. The response is a JSON object containing a list of device details, including the device name, IP address, and reachability status.

```
{
  "response": [
    {
      "family": "Switches and Hubs",
      "errorCode": null,
      "type": "Switches and Hubs",
      "apManagerInterfaceIp": "",
      "associatedWlCip": "",
      "bootDateTime": "2018-10-16 10:37:51",
      "collectionInterval": "00:00:00",
      "collectionStatus": "Partial Collection Failure",
      "errorCode": "CLI-AUTH-ERROR",
      "errorDescription": "CLI password for the dev family: 'Switches and Hubs'",
      "hostname": "cat_9k_1.marius.x-trem.ro",
      "id": "1a85db61-8bf2-4717-9060-9776f42e4581",
      "instanceTenantId": "5bd3634ab2bea0004c3ebb58",
      "instanceUuid": "1a85db61-8bf2-4717-9060-9776",
      "interfaceCount": "41",
      "inventoryStatusDetail": "<status><general co",
      "lastUpdateTime": "1556258592769",
      "lastUpdated": "2019-04-26 06:03:12",
      "lineCardCount": "2",
      "lineCardId": "df065d20-8d9b-4b66-a5ed-30aab5",
      "location": null,
      "locationName": null,
      "macAddress": "f8:7b:20:67:62:80",
      "managementIpAddress": "10.10.22.66",
      "memorySize": "889226872",
      "platformId": "C9300-24UX",
      "reachabilityFailureReason": "Collection Fail",
      "reachabilityStatus": "Unreachable",
      "role": "DISTRIBUTION",
      "roleSource": "MANUAL"
    }
  ]
}
```

Summing up

# What did we talk about?

- Don't be bashful, Love thy terminal
- Shhh... SSH like a pro
- Oscar worthy bash scripts!  
(and other bash-y stuff)
- Make your Apps Dance



# Webinar Resource List

- Docs and Links
  - <https://developer.cisco.com/python>
  - [Bash and Guestshell on NX-OS](#)
  - [Guest Shell on IOS XE](#)
- Learning Labs
  - Laptop Setup <http://cs.co/lab-dev-setup>
  - Introduction to Guest Shell on IOS XE <http://cs.co/lab-iosxe-guestshell>
- DevNet Sandboxes
  - Multi-IOS Sandbox with VIRT <http://cs.co/sbx-multi>
- Code Samples
  - [https://github.com/hpreston/netdevops\\_demos](https://github.com/hpreston/netdevops_demos) -> dev\_tips folder



# NetDevOps Live! Code Exchange Challenge

[developer.cisco.com/codeexchange](https://developer.cisco.com/codeexchange)

***Put together some bash scripts and/or a Makefile for one of your network automation projects to help setup the environment.***

*Example: Add "make dev" and "make clean" capabilities to your project.*

The image shows two overlapping screenshots of the Cisco Code Exchange website. The top screenshot is a search page with the heading "Discover code repositories related to Cisco technologies". It includes a search bar with the placeholder text "What are you looking for?". Below the search bar are four circular icons representing different technologies: a blue and white icon, a colorful circular icon, a white circle with the word "Kinetic", and a white circle with the word "Meraki". Below these icons are three dropdown menus: "Recommended", "Python", and "Networking". The bottom screenshot is the "Submit your project" form. It has a title "Submit your project" and a "Git Repo URL:" field with the placeholder "https://github.com/organization/repository". Below this is a section "Select all technologies associated with this repo:" with a list of checkboxes: IoT, Cloud, Networking, Data Center, Open Source, Collaboration, Analytic & Automation, Security, and Mobility. There are also two questions: "Do you own or have permission to submit this repo?" and "Is this a Cisco repo?". At the bottom of the form is a checkbox for "Agree to Terms and Conditions" and a blue "Submit project" button. To the right of the form is a large illustration of three people (two men and one woman) sitting around a laptop, with a GitHub logo and code symbols above them.



# Looking for more about NetDevOps?

- NetDevOps on DevNet  
[developer.cisco.com/netdevops](https://developer.cisco.com/netdevops)
- NetDevOps Live!  
[developer.cisco.com/netdevops/live](https://developer.cisco.com/netdevops/live)
- NetDevOps Blogs  
[blogs.cisco.com/tag/netdevops](https://blogs.cisco.com/tag/netdevops)
- Network Programmability Basics Video Course  
[developer.cisco.com/video/net-prog-basics/](https://developer.cisco.com/video/net-prog-basics/)



Got more questions? Stay in touch!



**Hank Preston**

 [hapresto@cisco.com](mailto:hapresto@cisco.com)

 [@hfpreston](https://twitter.com/hfpreston)

 <http://github.com/hpreston>



**[developer.cisco.com](https://developer.cisco.com)**

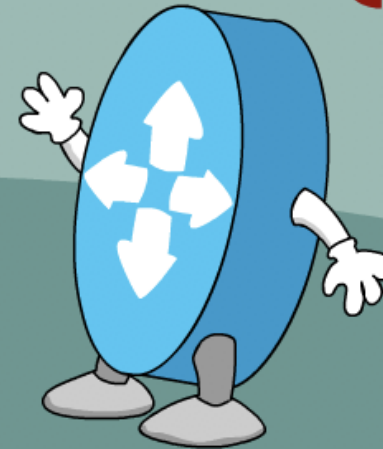
 [@CiscoDevNet](https://twitter.com/CiscoDevNet)

 [facebook.com/ciscocodevnet/](https://facebook.com/ciscocodevnet/)

 <http://github.com/CiscoDevNet>



# NETDEVOPS {LIVE!}



DEVNET

<https://developer.cisco.com/netdevops/live>

@netdevopslive 