Arista Network Devices Automation

Instructor Notes

Instructor: Daniel Ashitey

Version: 2.0
Date: November, 2023

Automation and NetOps

- **Automation**
  - Self-Service
  - Configuration by scripts
  - Multiple changes in a day
  - Very dynamic environment
  - Modern development practices
  - Faster time to market
- **Legacy**
  - Change Control
  - Manual Configuration by cli
  - Limited changes in a day
  - Environment is rigid
  - Prone to errors
  - Longer time to market
- **NetOps**
  - Infrastructure changes
  - Configuration management
  - Image management
  - Backups and restores
- **DevOps**
  - Practice of collaboration
  - Software development teams
  - Tools and processes
  - Automation

Key Components in Automation

- Programmable infrastructure
  - Controllers (Orchestration Managers)
  - Available APIs (REST API/ eAPI, NETCONF, YANG)
  - Structured data models (json, yaml, xml)
  - Scripting Language (Python, Golang, Ruby)
- Single Source of Truth (SSOT)
  - System implementation blueprint
  - Version Control
- Zero Touch Provisioning
  - Process of configuring objects without human intervention
  - Need dhcp options, tftp-server, config file
- Automation Tools
  - Configuration management – Ansible, Puppet, Chef SaltStack
  - Version Control - GitHub
  - Arista – CVP

Zero Touch Provisioning

- DHCP
  - Option 66 - Boot Server Host Name (TFTP)
  - Option 67 - Bootfile Name
    - http://192.168.0.24/ztp/bootstrap

- CVP
  - SYS_TelemetryBuilder
    - Configlet Builder
    - Builds ‘daemon TerminAttr’ config

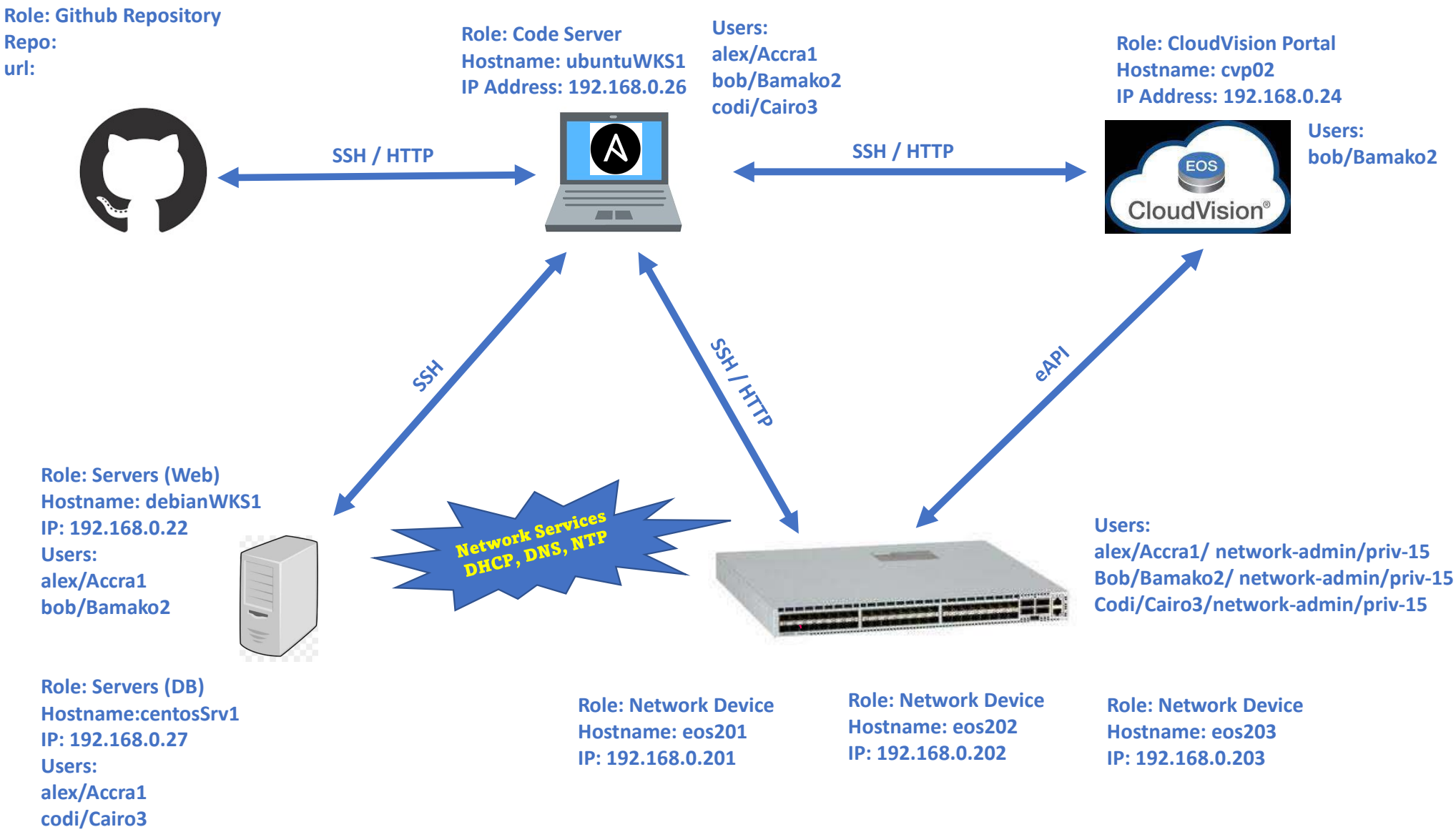
- DEMO

SSH

- User config on managed device
  - Local user:
    - password: secret
    - privilege: 15
    - role: network-admin
    - ssh-key
  - RADIUS User
- SSH-Key generation and transfer
  - passphrase
  - no passphrase
- SSH verification
  - known host
  - authenticated keys

Git

- Git Repository Setup
  - Github
- Clone a to local git repository
- Git commands
  - Git init
  - Git status
  - Git add
  - Git commit
  - Git push


Role: Github Repository
Repo:
url:
Role: Code Server
Hostname: ubuntuWKS1
IP Address: 192.168.0.26
Users:
alex/Accra1
bob/Bamako2
codi/Cairo3
Role: CloudVision Portal
Hostname: cvp02
IP Address: 192.168.0.24
Users:
bob/Bamako2
SSH / HTTP
SSH / HTTP
EOS
CloudVision
SSH
SSH / HTTP
eAPI
Role: Servers (Web)
Hostname: debianWKS1
IP: 192.168.0.22
Users:
alex/Accra1
bob/Bamako2
Network Services
DHCP, DNS, NTP
Users:
alex/Accra1/ network-admin/priv-15
Bob/Bamako2/ network-admin/priv-15
Codi/Cairo3/network-admin/priv-15
Role: Servers (DB)
Hostname:centosSrv1
IP: 192.168.0.27
Users:
alex/Accra1
codi/Cairo3
Role: Network Device
Hostname: eos201
IP: 192.168.0.201
Role: Network Device
Hostname: eos202
IP: 192.168.0.202
Role: Network Device
Hostname: eos203
IP: 192.168.0.203

Install SSH

```
alex@UbuntuWKS-1:~$ sudo apt  update && sudo apt upgrade
alex@UbuntuWKS-1:~$ sudo apt  install openssh-server
```

SSH (no user alex in remote server)

```
alex@UbuntuWKS-1:~$ ssh centosSrv1
The authenticity of host 'centossrv1 (192.168.0.27)' can't be established.
ECDSA key fingerprint is SHA256:Z30sE2u3879fiKmYiTvhT2CCOxTM76ihvGVNljzGsXA.
Are you sure you want to continue connecting (yes/no/[fingerprint])? Yes
Warning: Permanently added 'centossrv1' (ECDSA) to the list of known hosts.
alex@centossrv1's  password:
Permission denied, please try again.
alex@centossrv1's  password:
Permission denied, please try again.
alex@centossrv1's  password:
alex@centossrv1's : Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
alex@UbuntuWKS-1:~$
```

SSH (user in remote server)

```
bob@UbuntuWKS-1:~$ ssh debianwks1
The authenticity of host 'debianwks1 (192.168.0.22)' can't be established.
ECDSA key fingerprint is SHA256:G6kicuK4SDH7sn4sc76WsDyAAY/Fg4ZgOxjbOPPyv0o.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'debianwks1,192.168.0.22' (ECDSA) to the list of known hosts.
bob@debianwks1's password:
Linux debianWKS1 4.19.0-17-amd64 #1 SMP Debian 4.19.194-2 (2021-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Jul 13 13:13:38 2021 from 192.168.0.125
bob@debianWKS1:~$


alex@UbuntuWKS-1:~$ ssh codi@centossrv1
codi@centossrv1's password:
Activate the web console with: systemctl enable --now cockpit.socket

Last login: Tue Jul 13 14:58:24 2021 from 192.168.0.125
[codi@centOSSrv1 ~]$
```

Known Hosts

```
alex@UbuntuWKS-1:~$ cat .ssh/known_hosts
|1|SF7o/d4JOwa5CVQTAvgDRwudA5k=|yXZRVjArirkPVbsatRISA2W4UjE= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBC0/2hacuQfxGCzcyTiqKwjTUlyxLShC
qOPyUVvP35qdiivcm9IPkEa41RvwnbKwMHeQ6FhwRgyjg7LMwOl7zbU=
|1|c9Ld7q34ZDunJCwB1NQTcYpFit0=|A5xB+aRXO0j6esx3cSguBkZiP0U= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBC0/2hacuQfxGCzcyTiqKwjTUlyxLShC
qOPyUVvP35qdiivcm9IPkEa41RvwnbKwMHeQ6FhwRgyjg7LMwOl7zbU=


bob@UbuntuWKS-1:~$ cat .ssh/known_hosts
|1|awN5YSpivlsaBAZnR2SrLgpfpxg=|t026JNDzyhEenCpsox6qRSQOPxg= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJbZ73rt20v7dRt9lNID3Zq23ZJ+8k3Iz
S7NOIukLVWItLt48reImRmLOQnaK3B4rzS5vWmbuG8ryvwSGkuogEU=
|1|lRkBncxeuray9zdVtCWkge+bHaA=|cT4aa1GB/7qakGybtujzpDYtQO0= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBJbZ73rt20v7dRt9lNID3Zq23ZJ+8k3Iz
S7NOIukLVWItLt48reImRmLOQnaK3B4rzS5vWmbuG8ryvwSGkuogEU=
```

SSH to Arista Switches

```
alex@UbuntuWKS-1:~$ ssh eos201
The authenticity of host 'eos201 (192.168.0.201)' can't be established.
ECDSA key fingerprint is SHA256:zuUgPMTqnAa92TjqbzUJ7vEPbOf2nbMiNc+ihsP/r5M.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'eos201,192.168.0.201' (ECDSA) to the list of known hosts.
Password:
Last login: Wed Jul 14 12:19:32 2021 from 192.168.0.125
eos201>

alex@UbuntuWKS-1:~$ ssh eos202
The authenticity of host 'eos202 (192.168.0.202)' can't be established.
ECDSA key fingerprint is SHA256:Rl8sIyLhUES2xRGz0qMueHz7MejpxPGQ5qs/Koz5uC0.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'eos202,192.168.0.202' (ECDSA) to the list of known hosts.
Password:
eos202>

alex@UbuntuWKS-1:~$ ssh eos203
The authenticity of host 'eos203 (192.168.0.203)' can't be established.
ECDSA key fingerprint is SHA256:xYWz5qVEjEF6WqcsXV7Mtu1+8RzqLHJBKvdBsV0mqW4.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'eos203,192.168.0.203' (ECDSA) to the list of known hosts.
Password:
eos203>
```

Known Hosts

```
alex@UbuntuWKS-1:~$ cat .ssh/known_hosts
|1|SF7o/d4JOwa5CVQTAvgDRwudA5k=|yXZRVjArirkPVbsatRISA2W4UjE= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBC0/2hacuQfxGCzcyTiqKwjTUlyxLShCqOPyUVvP35qdiivcm9IPkEa41RvwnbKwMHeQ6Fhw
Rgyjg7LMwOl7zbU=
|1|c9Ld7q34ZDunJCwB1NQTcYpFit0=|A5xB+aRXO0j6esx3cSguBkZiP0U= ecdsa-sha2-nistp256
AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABBBC0/2hacuQfxGCzcyTiqKwjTUlyxLShCqOPyUVvP35qdiivcm9IPkEa41RvwnbKwMHeQ6Fhw
Rgyjg7LMwOl7zbU=
|1|BgWMTNJtWvBzN5cYyMrgTqVFfig=|gGyp9X1OU6o6ls8QTov10LY53Iw= ecdsa-sha2-nistp521
AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBACSSgxvgwmWPJtenZ3IqK3DlxI4PNYa0vGVG4cAAhk/yVUAEdtfO9toX/fxC0QhAMRsWIK
/BsDN0YwrxKCF2zcHDABXSIhHnFI5sk3MlegfGMhHZn9JVkcp2QzxQv5UcPV5mMRo/lvrCPq3i5ScAwXBEyBJFFOY7/jJTpn6+dekc2vehA==
|1|boePd506gHiIU2PSD0ecuJ8oV+Q=|o9pAIyDoh4EXRa1NOcFKNUqOMV4= ecdsa-sha2-nistp521
AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBACSSgxvgwmWPJtenZ3IqK3DlxI4PNYa0vGVG4cAAhk/yVUAEdtfO9toX/fxC0QhAMRsWIK
/BsDN0YwrxKCF2zcHDABXSIhHnFI5sk3MlegfGMhHZn9JVkcp2QzxQv5UcPV5mMRo/lvrCPq3i5ScAwXBEyBJFFOY7/jJTpn6+dekc2vehA==
|1|DEF60Bo2opuV84ECtg1qeJWioCc=|5SMPLig8CY7C4MbgOSChnCVPmz4= ecdsa-sha2-nistp521
AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAH17MSaGFbEk59RqmGCUAXaO4uJuhF2Prr5T6RJiESYz6YEk06pXLFCTtsSkLI6bXG9OLTz
OqUe/e8MH4knGZHeLgHpN5yKiwDe02FSWJelkUyU4n28A1WWnr5nwe+7dg68n6HcjS8zoQhQ8bw/lvYd0+bG7/Teedg/XTbnduhe3bxmfA==
|1|YyPkKlbF+vxBxYa0jtuAygeY5gw=|bGM/VQNRA15/iUpLqEDnwk0+VsI= ecdsa-sha2-nistp521
AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAH17MSaGFbEk59RqmGCUAXaO4uJuhF2Prr5T6RJiESYz6YEk06pXLFCTtsSkLI6bXG9OLTz
OqUe/e8MH4knGZHeLgHpN5yKiwDe02FSWJelkUyU4n28A1WWnr5nwe+7dg68n6HcjS8zoQhQ8bw/lvYd0+bG7/Teedg/XTbnduhe3bxmfA==
|1|hdeP+19eYZMuEbvptjZmakXS6t0=|UIrz8qupZDn1etsKnq+RMbASTv8= ecdsa-sha2-nistp521
AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAAYeeOUA7ni4GhQi0XpGqhzLg3BG58ZU0WrBVZTQkn8PqHiLMZGSyEfZCjnkugQjGP8cga
Te3V9xWIzH126nctJ8QELN6iF/d9qoNBhanEyJx7HLW+BfEo1fx+1qpD1IPnkfbIjk7mxJp+rP2vK2iels2Z6N4s+Mv8nBBgQ7eAFQXG76A==
|1|3nK9PkMbQltaxeGkakxhSqEExQo=|v4k127tCXRLNjK+hcpbkR+NSa2Y= ecdsa-sha2-nistp521
AAAAE2VjZHNhLXNoYTItbmlzdHA1MjEAAAAIbmlzdHA1MjEAAACFBAAYeeOUA7ni4GhQi0XpGqhzLg3BG58ZU0WrBVZTQkn8PqHiLMZGSyEfZCjnkugQjGP8cga
Te3V9xWIzH126nctJ8QELN6iF/d9qoNBhanEyJx7HLW+BfEo1fx+1qpD1IPnkfbIjk7mxJp+rP2vK2iels2Z6N4s+Mv8nBBgQ7eAFQXG76A==
```

SSH-Key generation

ssh-keygen –t rsa –C “default”

- Specify with or without passphrase
- Options
  - t – type [dsa | ecdsa | ecdsa-sk | ed25519 | ed25519-sk | rsa]
  - C - comment
- Output:
  - Key Fingerprint: SHA256.....
  - Private key file: id_rsa
  - Public key file: id_rsa.pub
  - Folder: ~/.ssh

SSH-Keygen with passphrase

```
alex@UbuntuWKS-1:~$ ssh-keygen -t ed25519 -C "default"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alex/.ssh/id_ed25519): /home/alex/.ssh/default
Enter passphrase (empty for no passphrase): xxxxxx
Enter same passphrase again:  xxxxxx
Your identification has been saved in /home/alex/.ssh/default
Your public key has been saved in /home/alex/.ssh/default.pub
The key fingerprint is:
SHA256:2m2ZjuJ/3iidw20baJKotd34Vn/7QUxfi36F0EMgotc default
The key's randomart image is:
+--[ED25519 256]--+
|       . . ...   |
|      . o . o    |
|      . . E . o..|
|       .     .+o+|
|        S    ..+o|
|       + o +o . .|
|      + +oB+.o o |
|     o.o.OBoo.o o|
|    ..oo=**oo. o+|
+----[SHA256]-----+
```

SSH-Keygen without passphrase

```
alex@UbuntuWKS-1:~$ ssh-keygen -t ed25519 -C "automate"
Generating public/private ed25519 key pair.
Enter file in which to save the key (/home/alex/.ssh/id_ed25519): /home/alex/.ssh/automate
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/alex/.ssh/automate
Your public key has been saved in /home/alex/.ssh/automate.pub
The key fingerprint is:
SHA256:310ibAFawx93uEH92EXBF5GP0zGlYF4OG2DRwkBPkZs automate
The key's randomart image is:
+--[ED25519 256]--+
|      .o+X*=.+BO|
|       *==oB+*=|
|       . .=+ooOB|
|        E...= =|
|       S   + ...|
|        . o o o |
|         . . .  |
|                |
|                |
+----[SHA256]-----+
```

SSH-Keygen Files

```
alex@UbuntuWKS-1:~$ ls .ssh
automate  automate.pub  default  default.pub  known_hosts

alex@UbuntuWKS-1:~$ cat .ssh/default
-----BEGIN OPENSSH PRIVATE KEY-----
[redacted]
-----END OPENSSH PRIVATE KEY-----

alex@UbuntuWKS-1:~$ cat .ssh/default.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBIL3nwhh6w5CjBtpS7CcCu945HfvC45TbCiBPF8cPus
default
```

SSH-Keygen Files

alex@UbuntuWKS-1:~$ ls .ssh
automate automate.pub default default.pub known_hosts

alex@UbuntuWKS-1:~$ cat .ssh/automate
-----BEGIN OPENSSH PRIVATE KEY-----
[redacted]
-----END OPENSSH PRIVATE KEY-----

alex@UbuntuWKS-1:~$ cat .ssh/automate.pub
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIICJ52iPKG08G0PDAMubdw4+SbQUk1w9Js8IS4MtvBZb
automate

Transfer SSH-Keys to remote device

Servers:

ssh-copy-id –i ~/.ssh/default.pub debianwks1

- Specify passphrase
- Options
  - i – input file

Network device:

username alex ssh-key [paste public key here]

Transfer SSH-Keys to remote server debianWKS1

```
alex@UbuntuWKS-1:~$ ssh-copy-id -i ~/.ssh/default.pub debianwks1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/alex/.ssh/default.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
alex@debianwks1's password: xxxxxxx

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'debianwks1'"
and check to make sure that only the key(s) you wanted were added.

alex@UbuntuWKS-1:~$ ssh debianwks1
alex@debianwks1's password:
Linux debianWKS1 4.19.0-17-amd64 #1 SMP Debian 4.19.194-2 (2021-06-21) x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
alex@debianWKS1:~$
```

Transfer SSH-Keys to remote server centossrv1

```
alex@UbuntuWKS-1:~$ ssh-copy-id -i ~/.ssh/default.pub centossrv1
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/home/alex/.ssh/default.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
alex@centossrv1's password:  xxxxxxx

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'centossrv1'"
and check to make sure that only the key(s) you wanted were added.

alex@UbuntuWKS-1:~$ ssh centossrv1
alex@centossrv1's password:

Activate the web console with: systemctl enable --now cockpit.socket

Last failed login: Wed Jul 14 11:13:08 EDT 2021 from 192.168.0.26 on ssh:notty
There were 14 failed login attempts since the last successful login.
[alex@centOSSrv1 ~]$
```

Authorized Keys

debianWKS1

```
alex@debianWKS1:~$ ls .ssh
authorized_keys

alex@debianWKS1:~$ cat .ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBIL3nwhh6w5CjBtpS7CcCu945HfvC45TbCiBPF8cPus default
```

centossrv1

```
[alex@centOSSrv1 ~]$ ls .ssh
authorized_keys
[alex@centOSSrv1 ~]$
[alex@centOSSrv1 ~]$ cat .ssh/authorized_keys
ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIBIL3nwhh6w5CjBtpS7CcCu945HfvC45TbCiBPF8cPus default
[alex@centOSSrv1 ~]$
```

Add SSH-Keys to network devices

eos201(config)# username alex ssh-key ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIICJ52iPKG08G0PDAMubdw4+SbQUk1w9Js8IS4MtvBZb automate

Verify:

eos201# sh users accounts
user: alex
role: network-admin
privilege level: 15
ssh public key: ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIICJ52iPKG08G0PDAMubdw4+SbQUk1w9Js8IS4MtvBZb automate

alex@UbuntuWKS-1:~$ ssh eos201
Password: ← password is still being prompted. Will fix next slide
Last login: Wed Jul 14 15:25:06 2021 from 192.168.0.125
eos201>

Use SSH-Agent to store passphrase

```
~$  eval “$(ssh-agent -s)”
Agent pid 2364

#check if agent is running
ps aux | grep 2323

# Add the pass phrase to the agent
~$  ssh-add ~/.ssh/automate
Enter passphrase:
Identity added: /home/danash/.ssh/id_rsa (default ssh)
```

Alias the commands

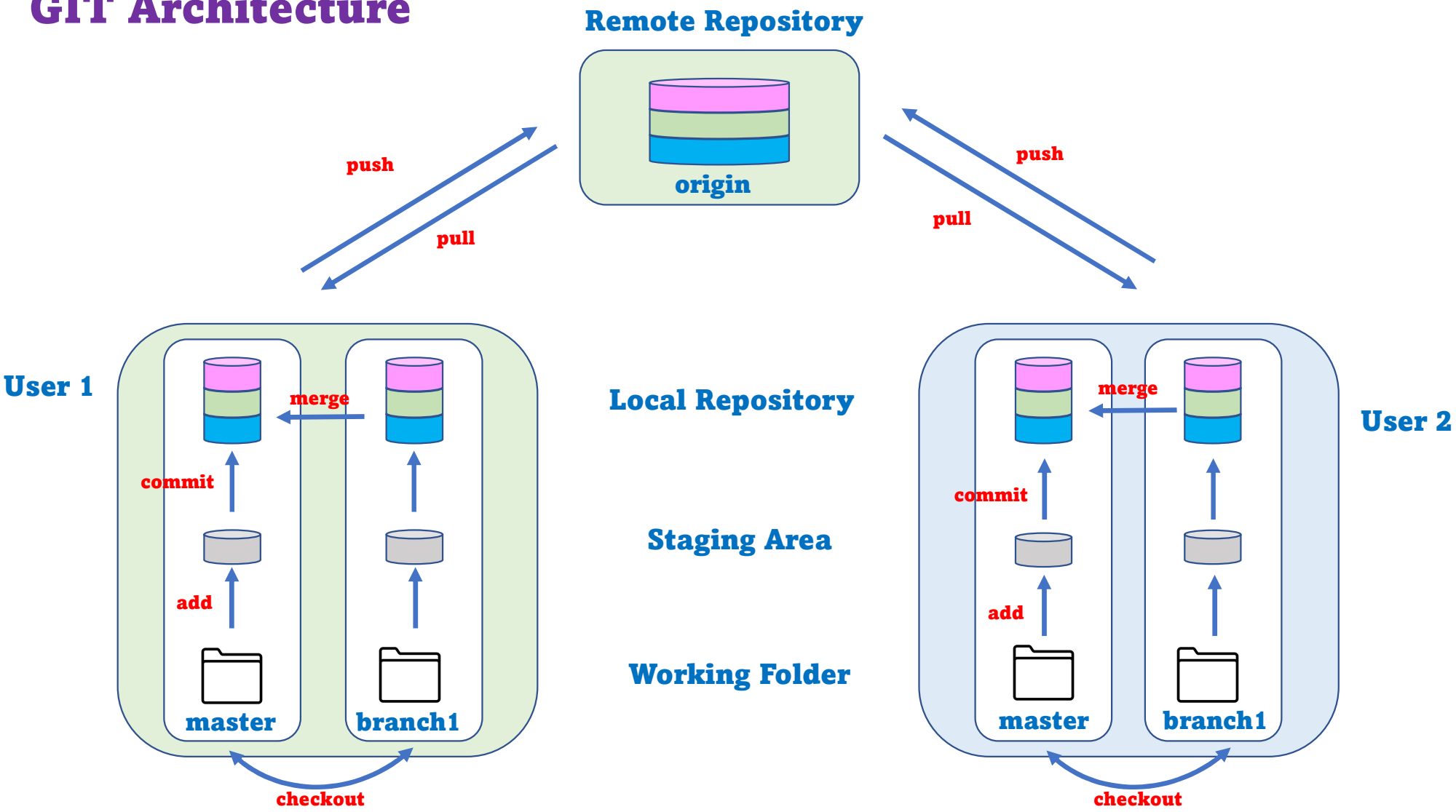
```
# include this in the .bashrc file to store the  alias
~$ nano .bashrc

# Alias
     alias ssha='eval$(ssh-agent -s) && ssh-add ~/.ssh/ automate’
# save and exit
```

GIT

- **Distributed version control system**
- **Tracks when and who made changes**
- **Separates workloads in main and branch folders**
- **Facilitate collaborations among multiple users**
- **Local and remote repositories**

GIT Architecture



GIT Install

- **Linux**
  sudo apt-get install git (Ubuntu, Debian)
  sudo yum install git (Fedora, CentOS)

- **MAC**
  http://git-scm.com/download/mac

- **Windows**
  http://git-scm.com/download/win

GIT Setup

- **Initialize project folder**

  ```
  myproject $ git init
  myproject (master) $ ls
  .git
  ```

- **Add name and email to git config**

  ```
  myproject (master)
  $ git config  --global user.name ‘alex’
  $ git config  --global user.email ‘alex@company.com’
  ```

  stores git config file: /etc/git.conf

GIT Commands

- **Add a file or files to the staging area**

  myproject (master) $ git add index.html
  myproject (master) $ git add *.html
  myproject (master) $ git add .

- **Remove file from staging area**

  myproject (master) $ git rm --cache index.html

- **Commit changes to local repository**

  myproject (master) $ git commit –m ‘add comments on change’

- **Ignore some files/folders from staging**

  myproject (master) $ nano .gitignore

  add files and/or folders in this file. They will be ignored by git add command

- **Change master to main**

  myproject (master) $ git branch –m main

GIT Branches

- Create a branch called branch1
  myproject (main) $ git branch branch1
  myproject (main) $

- Change working folder to branch 1
  myproject (main) $ git checkout branch1
  myproject (branch1) $

- Add a files to the branch1 staging area
  myproject (branch1) $ git add index.html
  myproject (branch1) $ git commit –m ‘add stuff in index’

- Change working folder back to master
  myproject (branch1) $ git checkout main
  myproject (main) $

- Merge the 2 local repositories
  myproject (main) $ git merge branch1 –m ‘added stuff in files’

GIT Remote Repository

- Create/Clone a repository in Github
  get the http url of the repo

- Clone a repository in Github
  myproject (main) $ git clone <<paste https url here>>

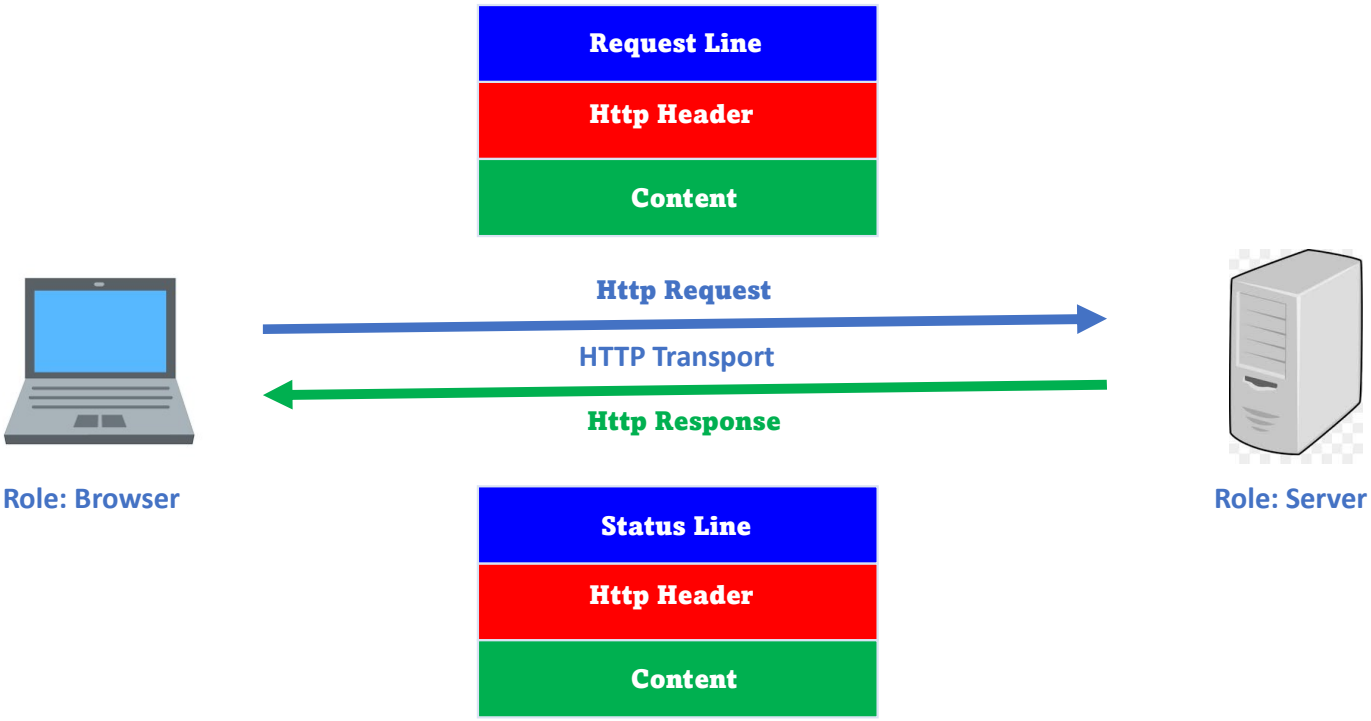
- Map/unmap the Github repo to the local repo
  myproject (main) $ git remote add origin <<paste https url here>>
  myproject (main) $ git remote rm origin
  myproject (main) $ git remote –v

origin

- Push local master repo to remote repo
  myproject (main) $ git push –u origin main
  prompt for login credentials. can be avoided by uploading ssh keys to github account

- Pull content from remote to local repo
  myproject (main) $ git pull

HTTP Basics


Request Line
Http Header
Content
Http Request
HTTP Transport
Http Response
Status Line
Http Header
Content
Role: Browser
Role: Server

HTTP Header

- **HTTP Request**

Request Line:

Method:
Path:
Protocol

Http Headers:

Host
User-Agent
Accept
Cookie
Referer
Content-Type
Content-Length
Authorization

Content

- **HTTP Response**

Status Line

Protocol:
Status Code:

Http headers:

Date
Server
Cookie
User-Agent
Content-Type
Content-Length

Content

HTTP Status Code

- **1xx: Informational**
  Request received / processing

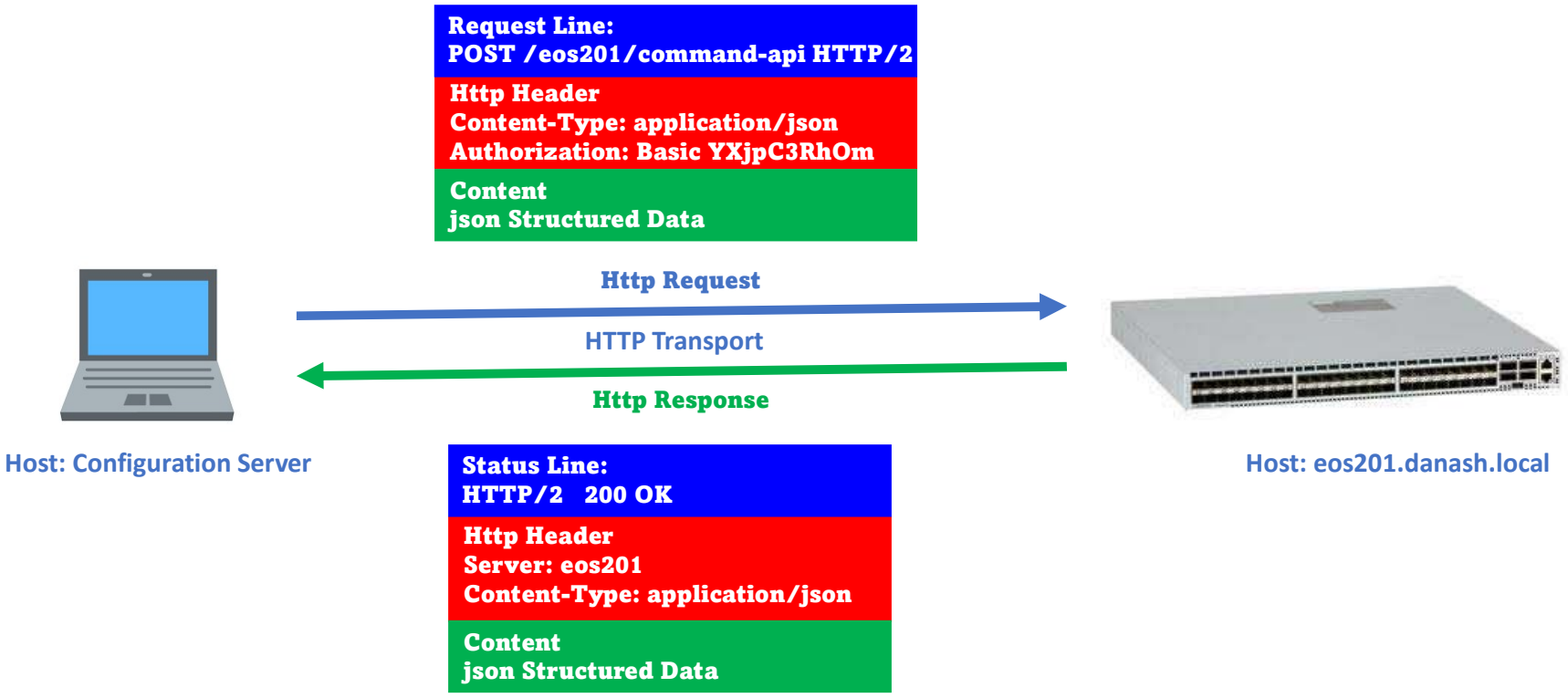
- **2xx: Success**
  Successfully received, understood and accepted
  e.g. 200 – OK

- **3xx: Redirect**
  Further action must be taken / redirect
  e.g. 301 – move to new url

- **4xx: Client Error**
  Request does not have what it needs
  404 – Not found

- **5xx: Server Error**
  Server failed to fulfill an apparent valid request
  502 – Bad gateway

Using HTTP to configure network devices



APIs

- Arista's eAPI with Web UI Explorer
  - Web UI Explorer
  - Script Editor

- REST API with Firefox RESTED Extension

- REST API with Chrome Postman
  - http request method: POST, GET, DELETE, PUT
  - database premitives: Create, Read, Update, Delete

- Run Curl with json structured data format
- Run Python with json structured data format
- Run Python with yaml structured data format
- Run Ansible with yaml structured data format

Scripting Language

Scripting Language concepts

- **Operational**
  - variables
  - addition, subtration, multiplication, division
  - boolean operation (and/or/xor), outcome (true/false)
  - functions, modules, packages

- **Data Types**
  - integers, strings, lists, tupples, sets, dictionary

- **Data Representation Format**
  - json, yaml, xml

- **Languages**
  - Interpreted - bash, python, perl
  - Compiled – c, Fortran, COBOL
  - Machine Language – x86, SPARC

JavaScript Object Notation - JSON

JSON Types

- **Null** null
- **Strings:** “Hello” “Daniel” “Toronto”
- **Numbers:** 1500 19.41 0.213
- **Booleans:** true false
- **Array:** [1, 3, 9, 27] [“Boston”, “London”, “Brisbane”]
- **Objects:** {“Name”: “Daniel”} {“City”: “Accra”}

JSON File

device.json

```
{
 "switch_id": 2121342,
 "switchname": "eos-201",
 "isModular": true,
 "modules": [{
     "slot1": [{
         "name": "DCS-7800-SUP",
          "type": "Supervisor",
         "interfaces": ["ma1"]
     }],
     "slot2": {
         "name": "7800R3-36P",
         "type": "Line Card",
         "interfaces": ["eth1/0", "eth1/1", "eth1/2", "eth1/3", "eth1/4"],
     }
  }]

}
```

JSON with HTML

```
<!DOCTYPE html>
<html>
<head>
<title>JSON Example</title>
</head>>
<body>
   <script type="text/javascript">
      let devices =
      {
            "switch _id": 2121342,
            "switchname": "eos-201",
            "isModular": true,
            "modules": [{
                  "slot1": {
                        "name": "DCS-7800-SUP",
                        "type": "Supervisor",
                        "interfaces": ["ma1"]
                  }},
                  {
                  "slot2": {
                        "name": "7800R3-36P",
                        "type": "Line Card",
                        "interfaces": ["eth1/0", "eth1/1", "eth1/2", "eth1/3", "eth1/4"],
                  }}
            ]

      }
     console.log(devices)
     console.log(devices.modules[1].slot2.interfaces[2])          ← verify with chrome dev tool
   </script>
</body>
</html>
```

JSON with Curl

```
$ curl -s -k -X POST \
  https://eos201/command-api \
   -H 'authorization: Basic YWxleDpBY2NyYTE=' \
   -H 'content-type: application/json' \
   -d '{
     "jsonrpc": "2.0",
     "method": "runCmds",
     "params": {
                "format": "json",
                "timestamps": false,
                "autoComplete": true,
                "expandAliases": false,
                "includeErrorDetail": false,
                "cmds": [
                   "show ver"
                 ],
                 "version": 1
     },
     "id": "EapiExplorer-1"
 }' | jq                    ← jq - Jason formatted on stout
```

curl options

-s silense or quiet mode
-k insecure
-X request
-H header
-d data
-u user credentials
-v verbose
-o output
-w write-out or display in stdout
@ read from a file or stdin
‘%{}’ string variable
Jq json processor

Python Basics

Variables

```
>>> print (“Hello”)
Hello

>>> x = “Hello”
>>> print (x)
Hello
```

Variable Naming

- Case Sensitive
- Must not start with a number
- Not allowed: $, %, @, -
- _ means private
- __ means very private
- __name__ means special identifier

Best Practise

- Avoid: is and,or, in not, …
- Don’t use all CAPs

Reference: w3schools.com/python

Python Basics – Data Types

- **Integers**

```
>>> x=10
>>> type(x)
<class 'int'>
```

- **Float**

```
>>> x=10.24
>>> type(x)
<class 'float'>
```

- **Complex**

```
>>> x=10.24j
>>> type(x)
<class 'complex'>
```

- **String - single quotes**

```
>>> x= 'Daniel'
>>> print(x)
Daniel
>>> type(x)
<class 'str'>
```

- **String - double quotes**

```
>>> x= "Daniel"
>>> print(x)
Daniel
>>> type(x)
<class 'str'>
```

- **String - triple quotes**

```
>>> x= "'Daniel'"
>>> print(x)
'Daniel'
>>> type(x)
<class 'str'>
```

- **String - triple quotes**

```
>>> x= '"Daniel"'
>>> print(x)
"Daniel"
>>> type(x)
<class 'str'>
```

Python Basics – String Data Types

- String – is a list of characters

```
>>> x= ‘“Daniel”’
>>> x[0]
‘“’
>>> x[4]
‘i’
```

- String – joining strings together

```
>>> x= ‘Daniel’ + ‘ ’ + ‘Ashitey’
>>> print(x)
Daniel Ashitey
```

- String – Length

```
>>> x= ‘“Daniel”’
>>> len(x)
8
```

- String – and special characters

```
>>> x= “backslash is \”
>>> print(x)
SyntaxError:

>>> x= “backslash is ‘\’”
>>> print(x)
backslash is ‘’

>>> x= “backslash is \\”
>>> print(x)
backslash is \
```

Python Basics - String/Integer Data Types Conversion

- **Strings and Integers**

```
>>> x = “My Year of graduation is ”
>>> year = 2020
>>> x + year
TypeError: can only concatenate str
(not "int") to str
```

- **joining strings together**

```
>>> x = “My Year of graduation is ”
>>> year = 2020
>>> x + str(year)
'My year of graduation is 2020’
```

- **Integers and Strings**

```
>>> my_age = 25
>>> your_age = “35”
>>> my_age + your_age
TypeError: unsupported operand type(s)
for +: 'int' and 'str’
```

- **joining integers together**

```
>>> my_age = 25
>>> your_age = “35”
>>> my_age + int(your_age)
60
```

Python Basics - List Data Types

- **List – comma separated elements in [ ]**

```
>>> colors=['Orange', 'Yellow', 'Green', 'Red', 'Pink', 'Black']
>>> type(colors)
<class 'list'>
>>> len(colors)
6
>>> colors[0]
'Orange'
>>> colors[2:5]
['Green', 'Red', 'Pink']
>>> sorted(colors)
['Black', 'Green', 'Orange', 'Pink', 'Red', 'Yellow']
>>> colors[1]='Blue'
>>> colors
['Orange', 'Blue', 'Green', 'Red', 'Pink', 'Black']
```

← mutable

Python Basics - Tuple Data Types

- **Tuple – comma separated elements in ( )**

```
>>> kitchenitems=('Blender', 'Stove', 'Refrigirator', 'Oven', 'Cookware', 'Cutlery')
>>> type(kitchenitems)
<class 'tuple'>
>>> len(kitchenitems)
6
>>> kitchenitems[0]
'Blender'
>>> kitchenitems[2:5]
('Refrigirator', 'Oven', 'Cookware')
>>> sorted(kitchenitems)
['Blender', 'Cookware', 'Cutlery', 'Oven', 'Refrigirator', 'Stove']
>>> kitchenitems[1]='Saucepan'
TypeError: 'tuple' object does not support item assignment    ← immutable
>>> x=list(kitchenitems)
>>> type(x)
<class 'list'>
>>> x
['Blender', 'Stove', 'Refrigirator', 'Oven', 'Cookware', 'Cutlery']
```

Python Basics - Set Data Types

- Set – comma separated elements in { }

```
>>> footballers={"Messi", "Gundogan", "Rudiger", " Haaland", "Mbappe", "Onana"}
>>> type(footballers)
<class 'set'>
>>> len(footballers)
6
>>> footballers[0]
TypeError: 'set' object is not subscriptable
>>> print("Messi" in footballers)
True
>>> sorted(footballers)
['Gundogan', 'Haaland', 'Mbappe', 'Messi', 'Onana', 'Rudiger']
>>> footballers[4]="Rice"
TypeError: 'set' object does not support item assignment    ← immutable
>>> y=list(footballers)
>>> type(y)
<class 'list'>
>>> y
['Gundogan', 'Haaland', 'Onana', 'Mbappe', 'Messi', 'Rudiger']
```

Python Basics - Dictionary Data Types

- Dict – comma sepatared key value pairs in { }

```
>>> device = {"hostname":"leaf1", "mac":"00505605ada7", "type":"veos"}
>>> type(device)
<class 'dict'>
>>> len(device)
3
>>> device[0]
KeyError: 0
>>> device["mac"]
'00505605ada7'
>>> sorted(device)
['hostname', 'mac', 'type']
>>> device["mac"]="005056d87f93"
>>> device
{'hostname': 'leaf1', 'mac': '005056d87f93', 'type': 'veos'}
>>> y=list(device)
>>> type(y)
<class 'list'>
>>>y
['hostname', 'mac', 'type']
```

Python Basics - Computation

Operators

```
>>>10+10
20
>>>100-40
60
>>>10*2
5
>>>30/3
10
>>> 5%2
1
>>> 2**5
32
```

Boolean operation

```
>>> 4<=2
False
>>> 10>8
True
```

Python Basics – Conditional/Loop Statement

- **IF**

```
nums=[5, 12, 22]
for x in nums:
  if x<=10:
    print(x, " is low")
  elif  x<=20:
    print(x, " is med")
  else:
    print(x, " is high")
```

- **While**

```
i=0
while i < 10:
  i += 1
  print(i)
else
print(i, “is not less than 10”)
```

- **For**

```
car=[‘Toyota’, ‘VW’, ‘GMC’. ‘Volvo’]
for x in car:
  print(x)
```

Python Basics - Functions

A code that performs specific task and returns a result

- **Built-in Functions**

  len() - returns the length of an object
  open() - opens a file and retuens a file object
  print() - prints to the stdout
  str() - returns a string object
  type() - returns the type of an object

- **User Defined Functions**

```
def function_name(param1, param2):
    operation statement uses param1 and/or param2
    built-in functions uses param1 and/or param2
    return (or not)

function_name(arg1, arg2)
```

Python Basics - Modules

A python file that contains a set of functions and or classes for performing various tasks.

- **Built-in Modules**

  datetime - contains functions to return information about the date object
  time - contains functions to return information about the time object
  sys - " ... system
  os - " ... operating system

- **Create a module**

```
def function_name(param1, param2):
    operation statement uses param1 and/or param2
    built-in functions uses param1 and/or param2
    return (or not)
```

  Save as MyFirstModule.py

- **Use a module**

```
Import MyFistModule.py
```

Python Basics - Modules

Location of Python Modules

Python performs a search on certain PATHs identified in the sys.path module.

- **Find the location of python a module**

```
>>> import sys
>>> dir(sys)
[... '__doc__', ... '__name__', '__package__', ... 'executable', 'exit', ... 'modules', 'path',
'path_hooks', ... 'stdin', 'stdout', ... 'version', ...]
>>>
>>>print(sys.path)
```

Windows:

```
['', 'C:\\Program Files\\Python39\\python39.zip', 'C:\\Program
Files\\Python39\\DLLs', 'C:\\Program Files\\Python39\\lib', 'C:\\Program
Files\\Python39', 'C:\\Program Files\\Python39\\lib\\site-packages']
```

Linux:

```
['', '/usr/lib/python39.zip', '/usr/lib/python3.9', '/usr/lib/python3.9/lib-dynload',
'/home/alex/.local/lib/python3.9/site-packages', '/usr/local/lib/python3.9/dist-
packages', '/usr/lib/python3/dist-packages', '/usr/lib/python3.9/dist-packages']
>>>
```

Python Basics - Package

A directory containing modules

- **Useful Packages**

  request - contains http libraries
  json - contains json libraries
  ansible - contains modules for functions relating to ansible
  yaml - contains modules for functions relating to yaml

- **Create a package**

  Create a folder called MyFirstPackage
  Add the modules in the directory
  Create a file called __INIT__ and list the modules in the file. Include functions as well.

- **Use a pachage**

  Import MyFirstModule
  MyFirstPackage.MyFirstModule.function_name(x1, x2)

Python Basics – I/O

Open Built-In Function

```
$ ls ~/project/alexL5Labs
mytext
>>> f = open('mytext', 'r')      {‘r’, ‘a’, ‘w’, ‘r+’, ‘a+’, ‘rt’, ‘rb’, ‘wt’, ‘wb’}
>>> f
<_io.TextIOWrapper name='mytext' mode='r' encoding='cp1252’>      ← windows
<_io.TextIOWrapper name='mytext' mode='r' encoding=‘UTF-8’>       ← Linux
>>>f.name
‘mytext’
>>>data = f.read()     {f.readline(), f.readlines(), write(str), seek(offset,origin), tell(), close()}
'1. eos201\n2. eos202\n3. eos203\n\n’
>>>data = f.read()
‘’
>>>f.tell()
31
>>>f.seek(10)
10
>>>data = f.read()
'2. eos202\n3. eos203\n\n’
>>>data = f.close()
>>>
```

```
>>> f=open('mytext')
>>>
>>> f.read(10)
'1. eos201\n'
>>>
>>> f.read(10)
'2. eos202\n'
>>> f.read(10)
'3. eos203\n'
>>> f.read(10)
'\n'
>>> f.read(10)
```

Python Basics – I/O

Open Built-In Function

Recommended approach using a context manager

```
>>> with open('mytext') as f:
...         print(f.read())
...
1. eos201
2. eos202
3. eos203
>>>
>>> with open('mytext') as f:
...         for line in f:
...             print(line, end='')
...
1. eos201
2. eos202
3. eos203
```

```
>>> with open('mytext') as f:
...             size = 4
...             data = f.readline(size)
...             while len(data) > 0:
...                         print(data, end='')
...                         data = f.readline(size)
...
1. eos201
2. eos202
3. eos203
```

Python Basics – I/O

OS Module - Manipulating folders in Python

```
>>> import os
>>>
>>> os.getcwd()
/home/alex
>>>
>>> os.chdir('project/alexL5Labs)
>>> os.getcwd()
/home/alex/project/alexLeLabs
>>>
>>> os.listdir()
mytext
>>> os.rename('mytext', 'yourtext')
>>> os.listdir()
yourtext
>>>
>>> os.copy('yourtext', 'mytext')
>>> os.listdir()
mytext yourtext
```

```
>>> os.remove('yourtext')
>>> os.listdir()
mytext
>>>
>>> os.mkdir('myfolder')
>>> os.listdir()
mytext myfolder
>>>
>>> os.rmdir('myfolder')
>>> os.listdir()
mytext
```

Reading text file converting to YAML and JSON files

Input file:
file1.text

Output file
Jason filename: file1.json

```
import json

with open('file1', 'r') as f:
      datafile = (f.read())

with open('file1.json', 'w') as outfile:
   json.dump(datafile, outfile)
```

Input file:
file2.text

Output file
Yaml filename: file2.yml

```
import yaml

with open('file2', 'r') as s:
      datafile = yaml.safe_load(s)


with open('file2.yml', 'w') as outfile:
   yaml.dump(datafile, outfile)
```

Convert JSON to python dict variable manually

```
>>> x={}
>>> x['switch_id'] = 2121342
>>> x["switchname"] = "eos-201“
>>> x["isModular"] = True
>>> x["modules"] = {"slot1":{"name": "DCS-7800-
SUP", "type": "Supervisor", "interfaces": ["ma1"]},
"slot2": {"name": "7800R3-36P", "type": "Line Card",
"interfaces": ["eth1/0", "eth1/1", "eth1/2",
"eth1/3", "eth1/4"]}}
>>>
>>>
>>>
>>> x["switchname"]
'eos-201'
>>>
>>> x["modules"][1]["slot2"]["interfaces"][4]
'eth1/4'
```

device.json

```
{
 "switch_id": 2121342,
 "switchname": "eos-201",
 "isModular": true,
 "modules": [{
      "slot1": [{
           "name": "DCS-7800-SUP",
            "type": "Supervisor",
           "interfaces": ["ma1"]
      }],
      "slot2": {
           "name": "7800R3-36P",
           "type": "Line Card",
           "interfaces": ["eth1/0", "eth1/1",
"eth1/2", "eth1/3", "eth1/4"]
      }
  }]

}
```

Convert JSON to python using json built-in package

```
>>> import json
>>> y = open('device.json')
>>> x = json.load(y)

>>> type(x)
<class 'dict'>
>>>
>>> x["switchname"]
'eos-201'
>>>
>>> x["modules"][1]["slot2"]["interfaces"][4]
'eth1/4'
```

```
import json

with open('device.json', 'r') as y:
      datafile = json.load(y)


with open('device.text', 'w') as d:
      print(datafile, file=d)
```

device.json

```
{
   "switch_id": 2121342,
   "switchname": "eos-201",
   "isModular": true,
   "modules": {
       "slot1": {
          "name": "DCS-7800-SUP",
          "type": "Supervisor",
           "interfaces": ["ma1"]
       },

       "slot2": {
            "name": "7800R3-36P",
            "type": "Line Card",
            "interfaces": ["eth1/0",
"eth1/1", "eth1/2", "eth1/3", "eth1/4"]
       }}


 }
```

JSON File

showver.json

```
'{
    "jsonrpc": "2.0",
    "method": "runCmds",
    "params": {
                "format": "json",
                "timestamps": false,
                "autoComplete": true,
                "expandAliases": false,
                "includeErrorDetail": false,
                "cmds": [
                   "show ver"
                 ],
                 "version": 1
    },
    "id": "EapiExplorer-1"
 }'
```

← Can be obtained through the EOS WEB UI

From Github to JSON and YAML File

```
Import request

r =
request.get(https://raw.githubusercontent.com/dashitey/alexL5Labs/main/int
erfaces)

devices-j = json.load(r)
devices-y = yaml.safe_load(r)
Print(devices-j)
Print(devices-y)
```

JSON with Python using HTTP client module

```
import http.client, ssl

conn = http.client.HTTPSConnection("eos201", context=ssl._create_unverified_context())

payload = "{\n  \"jsonrpc\": \"2.0\",\n  \"method\": \"runCmds\",\n  \"params\": {\n
\"format\": \"json\",\n    \"timestamps\": false,\n    \"autoComplete\": true,\n
\"expandAliases\": false,\n    \"includeErrorDetail\": false,\n    \"cmds\": [\n      \"show
ver\"\n    ],\n    \"version\": 1\n  },\n  \"id\": \"EapiExplorer-1\"\n}"

headers = {
    'content-type': "application/json",
    'authorization': "Basic YWxleDpBY2NyYTE="
       }

conn.request("POST", "/command-api", payload, headers)

res = conn.getresponse()
data = res.read()

print(data.decode("utf-8"))
```

JSON with Python using HTTP requests module

```
import requests
import json

url = "https://eos201/command-api"

with open("showver2.json", "r") as f:
        commands = f.read()

r = requests.post(url, auth=('alex','Accra1'), data=commands, verify=False)

print(r.text)
```

CVP SSOT

Single Source of truth

- CVP Configlet
  - Configlet Store
- Python Scripts
  - Code Server
  - CVP Configlet Builder and Forms
- Ansible direct
  - Code Server
- Ansible via CVP
  - Code Server

JSON with Python using cvp API

```
import requests
import json

from requests.packages.urllib3.exceptions import InsecureRequestWarning
requests.packages.urllib3.disable_warnings(InsecureRequestWarning)

cvp_host = "cvp02"
cvp_user = "bob"
cvp_pass = "Bamako2"

url = "https://%s"%cvp_host
headers = { 'content-type': 'application/json' }
loginURL = "/web/login/authenticate.do"
authenticateData = json.dumps({'userId': cvp_user, 'password': cvp_pass})
response = requests.post(url+loginURL, data=authenticateData, headers=headers, verify=False)
assert response.ok
cookies = response.cookies
output = response.json()

print("User Name: %s\n "%output['username'])
print("First Name: %s\n "%output['user']['firstName'])
print("Last Name: %s\n "%output['user']['lastName'])
print("User Permissions: %s\n "%output['permissionList'])
print("Cookie Jar: %s\n "%cookies)
```

CVP Python Executions Environment

CVP-specific python libraries that provides access to the various CVP services and device state

- **cvplibrary package**
  - **Form**
  - **CVPGlobalVariables**
  - **CVPGlobalNames**
  - **Example configlets**
    - **[root@cvp02 ~]# cd /cvpi/tools**
    - **[root@cvp02 tools]# ./cvptool.py --host cvp02 --user bob --password Bamako2 --objects Configlets --action restore --tarFile examples.tar.**

- **arista-cvp-scripts (based on cvprac libraries)**
  - **cvp-container-manager**
  - **cvp-configlet-manager**
  - **cvp-task-manager**
  - **cvp-configlet-backup**

Example – Using Arista cvplibrary in python script

```
from cvplibrary import CVPGlobalVariables, GlobalVariableNames, Device

def create_routes(hostname):
   number = hostname[-1:]

   for x in range(201, 204):
            if hostname.endswith(str(x)):
                        print "interface ethernet 5"
                        print "   no switchport"
                        print "   ip add 10.10.10.%d/31" % (x)
   return
user = CVPGlobalVariables.getValue(GlobalVariableNames.CVP_USERNAME )
passwd = CVPGlobalVariables.getValue(GlobalVariableNames.CVP_USERNAME )
ip = CVPGlobalVariables.getValue(GlobalVariableNames.CVP_IP )

ss = Device(ip)

def get_hostname():
   show_hostname = ss.runCmds(["enable", "show hostname"])[1]
   hostname = show_hostname['response']['hostname']
   return hostname

def main():
   hostname = get_hostname()
   if hostname.startswith("eos"):
     create_routes(hostname)

if __name__ == "__main__":
   main()
```

WiFi API

Arista Wi-Fi Services REST API Documentation

Wireless Manager modules

- **session** - session related operations, like create a new session(authenticate) and delete a session (logout).
- **locations** - all operations to fetch and manage the location hierarchy.
- **manageddevices** - operations to fetch and manage the Devices managed by the Wireless Manager.
- **aps**- fetch and manage the Access Points managed and detected by the Wireless Manager.
- **clients** - fetch and manage the Client devices associated and/or visible to the Managed Devices.
- **configuration** - comprises of all configuration policy related operations, like fetching or modifying a policy at any location.
- **deviceconfiguration** - fetch and manage all kinds of Template and Profile configuration objects like Device Template, SSID Profile, Role Profile, etc.
- **troubleshooting** - comprises of all the device troubleshooting related operations, like initiating a client connectivity test or a packet capture.
- **modules** - fetch all the supported modules and their versions.

Guest Manager modules

- **System Calls** - Contains APIs for system operations such as sign in, sign out.
- **Analytics** - Contains APIs to fetch time and location-based analytical data.
- **Dashboard** - Contains APIs to fetch demographic data about guest profiles.
- **Reports** - Contains APIs to perform different operations on reports such as creating, viewing, deleting, or emailing a report.
- **Location** - Contains APIs to fetch SSIDs and the location tree.
- **Servers** - Contains APIs to add, delete or update a server, retrieve a list of servers or retrieve more information about a particular server.
- **Portals** - Contains APIs to configure portals for authenticating the guests.
- **Plug-ins** - Contains APIs to retrieve and edit plug-in configurations

https://apihelp.wifi.arista.com

ANSIBLE

Ansible

- Open Source IT Tool
  - Orchestration
  - Automation
  - Cloud Provisioning
  - Configuration Management
  - Multi-Tier deployment
  - Many more…

- Key drivers
  - Agentless
  - Push model
  - Idempotent

Ansible Setup

- **Install Ansible**
  - **sudo apt update && sudo apt upgrade**
  - **sudo apt install ansible**

- **Create git repository folder for ansible projects**
  - **~/projects/alexL5Labs**
  - **Git to remote repo (https://github.com/dashitey/alexL5Labs)**

- **Create config and inventory files**
  - **~/projects/alexL5Labs**
  - **ansible.cfg ← e.g. (https://github.com/ansible/ansible/blob/devel/examples/ansible.cfg)**
  - **inventory**

- **Add ssh-keys to eos devices**
  - **username alex ssh-key <<automate alex>>**

- **Run Test on inventory to verify reachability to host**
  - **Ansible all –i inventory –m ping**

Ansible Components

- SSOT
  - the system running ansible:
    coderWKS
- Nodes
  - the switches and servers that need to be configured:
    leaf1, leaf2, leaf3, leaf4
- Inventory File
  - Contains individual and groups of nodes to be configured:
    inventory.yml
- Playbook
  - Yaml file containing one or more plays. A play is a task to be performed on a host. Rely on ansible modules and variables. Simplifies using of jinja templates.
    playbook.yml, ansible and arista modules
- Templates
  - Host configuration files that use variables instead of actual values. Can be applied to multiple hosts.
    Vlan.j2
- Var
  - Files containing variables referenced in playbooks:
    CVPmodel.yml

Ansible Connection to Arista devices

- **Via Http**

  In the inventory – [all:var] section add:

  ansible_connection = httpapi
  ansible_httpapi_port = 443
  ansible_httpapi_use_ssl = True
  ansible_httpapi_validate_certs = False
  ansible_network_os = eos
  ansible_become = yes
  ansible_become_method = enable
  ansible_python_interpreter = /usr/bin/python3
  ansible_user = alex
  ansible_password = Accra1

- **Via SSH**

  In the inventory – [all:var] section add:

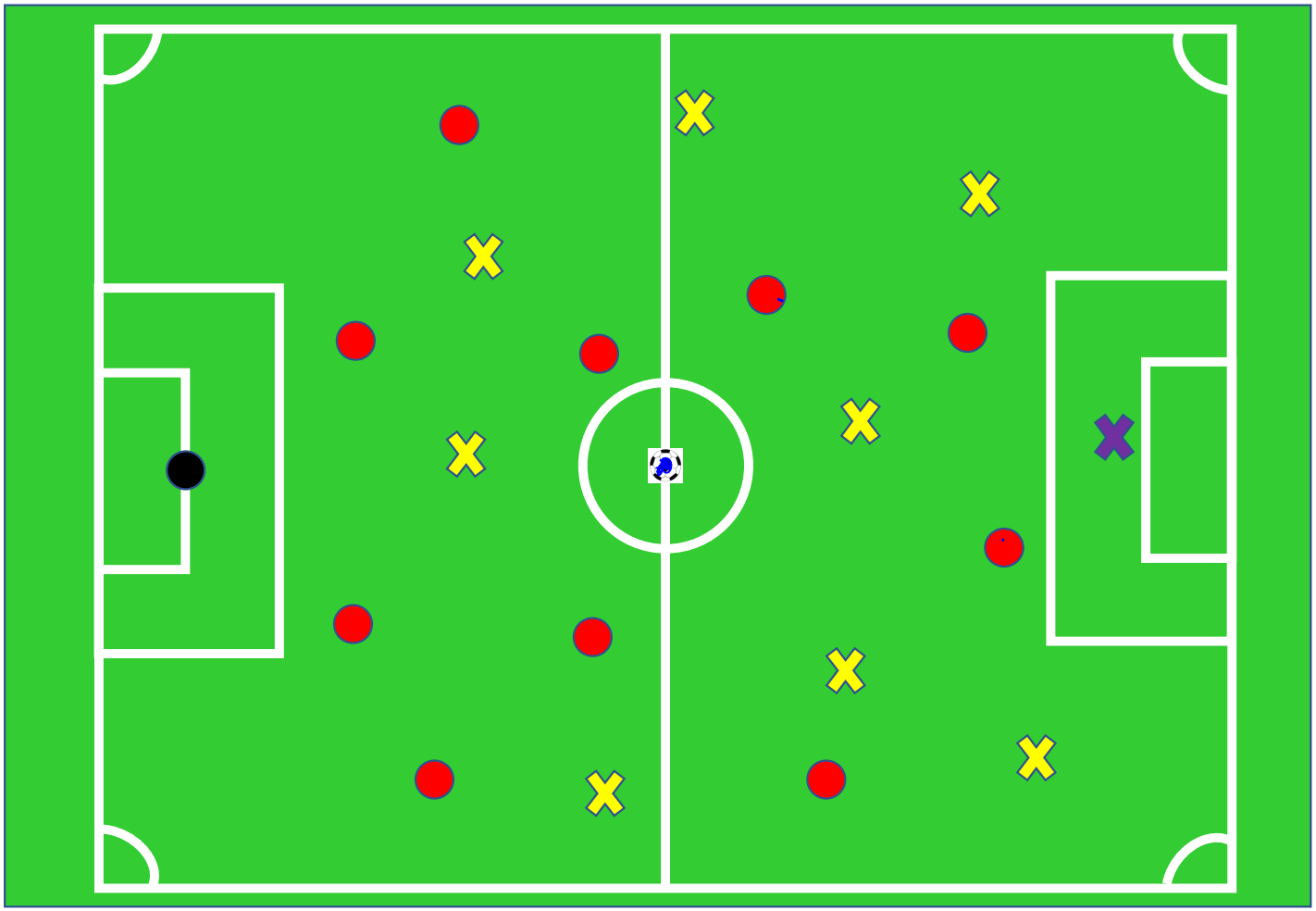
  ansible_connection = network_cli
  ansible_network_os = eos
  ansible_become = yes
  ansible_become_method = enable
  ansible_python_interpreter = /usr/bin/python3
  ansible_user = alex

Ansible Playbook

Config
- Pitch: green
- Goal_post_size: 10x17
- Ball: inflated
- Referee: Qualified

Inventory
- Var:
  - Jersey (color, num)
- Host:
  - Players

Playbook
- Play1: 4-2-4
  - task
- Play2: 4-4-2
  - task
- Play3: 4-3-2-1
  - task



Playbook - Soccer

Yet Another Markup Language - YAML

YAML Format

- Line separation
- Indentation
- Key value pairs

YAML Types

- Null None
- Strings: Hello ‘Daniel’ “Toronto”
- Numbers: 1500 19.41 0.213
- Booleans: true false yes no
- List: -
- Objects: Indentation

YAML File

Objects

```
eosdevice:
    switch_id: 2121341
    switchname: eos201
    isModular: yes
```

List of Objects

```
eosdevice:
   - switch_id: 2121341
     switchname: eos201
     isModular: yes
   - switch_id: 3427611
     switchname: eos202
     isModular: no
```

Lists within list

```
eosdevice:
   - switch_id: 2121341
     switchname: eos201
     isModular: yes
     modules:
     - slot1:
          name: "DCS-7800-SUP"
          type: "Supervisor",
          interfaces: ["ma1"]
     - slot2:
          name: ‘7800R3-36P
          type": Line Card
          interfaces:
          - eth1/0
          - eth1/1
          - eth1/2
          - eth1/3
          - eth1/4
```

Convert yaml to python dict using yaml package

Install yaml package
~$ pip install pyyaml

```
import yaml

x =yaml.load(open('device.yml'), Loader=yaml.SafeLoader)

print(x[0]['switchname'])

print(x[0]["modules"]["slot2"]["interfaces"][4])

print(x[1]["interfaces"][0])
```

device.yml

```
- switch_id: 2121341
  switchname: eos201
  isModular: yes
  modules:
    slot1:
      name: "DCS-7800-SUP"
      type: "Supervisor"
      interfaces: ["ma1"]
    slot2:
      name: '7800R3-36P'
      type": Line Card
      interfaces:
        - eth1/0
        - eth1/1
        - eth1/2
        - eth1/3
        - eth1/4
- switch_id: 3427611
  switchname: eos202
  isModular: no
  interfaces: [eth1/0, eth1/1]
```

Ansible-Galaxy Collections

- Install arista collections (eos, cvp, avd)
  - ansible-galaxy collection install arista.eos
  - ansible-galaxy collection install arista.cvp
  - ansible-galaxy collection install arista.avd

- Install ansible.netcommon
  - ansible-galaxy collection install ansible.netcommon

- List installed ansible galaxy collections
  - ansible-galaxy collection list

Info on Arista Automation Tools

- **Arista Ansible Collection**
  https://github.com/aristanetworks/ansible-eos
  https://github.com/aristanetworks/ansible-cvp
  https://github.com/aristanetworks/ansible-avd

- **Arista Validated Design**
  https://avd.sh/en/stable/docs/getting-started/intro-to-ansible-and-avd.html

- **CVPRAC**
  https://github.com/aristanetworks/cvprac
  https://pypi.org/project/cvprac

- **PyeAPI**
  https://pypi.org/project/pyeapi/
  https://pyeapi.readthedocs.io/en/latest/

Ansible Modules - Arista eos modules

~/.ansible/collections/ansible_collections/arista/eos/plugins/modules

- **eos_acls.py** : ACLs resource module
- **eos_banner.py** : Manage multi-line banners on Arista EOS devices
- **eos_bgp_address_family.py** : Manages BGP address family resource module
- **eos_bgp_global.py** : Manages BGP global resource module
- **eos_command.py** : Run arbitrary commands on an Arista EOS device
- **eos_config.py** : Manage Arista EOS configuration section
- **eos_eapi.py** : Manage and configure Arista EOS eAPI
- **eos_facts.py** : Collect facts from remote devices running Arista EOS
- **eos_interfaces.py** : Interfaces resource module
- **eos_l2_interfaces.py** : L2 interfaces resource module
- **eos_l3_interfaces.py** : L3 interfaces resource module
- **eos_lag_interfaces.py** : LAG interfaces resource module
- **eos_lldp.py** : Manage LLDP configuration on Arista EOS network devices
- **eos_logging.py** : Manage logging on network devices
- **eos_ospfv2.py** : Manages the attributes of ospfv2 on Arista EOS
- **eos_ospf_interfaces.py** : OSPF Interfaces Resource Module.
- **eos_static_route.py** : Manages the attributes of static routes on Arista EOS
- **eos_system.py** : Manage the system attributes on Arista EOS devices
- **eos_user.py** : Manage the collection of local users on EOS devices
- **eos_vlans.py** : VLANs resource module
- **eos_vrf.py** : Manage VRFs on Arista EOS network devices

Ansible Modules - Arista cvp modules

~/.ansible/collections/ansible_collections/arista/cvp/plugins/modules

- **cv_configlet.py :**
- **cv_configlet_v3.py**
- **cv_container.py**
- **cv_container_v3.py**
- **cv_device.py**
- **cv_device_v3.py**
- **cv_facts.py**
- **cv_task.py**
- **cv_task_v3.py**

Additional Packages for CVP connectivity

- **Install cvprac**
  - **pip install cvprac**
- **Install jsonschema**
  - **pip install jsonschema**

cvprac

This module provides a RESTful API client for Cloudvision® Portal (CVP) which can be used for building applications that work with Arista CV

- **cvp_api.py**
  - connect
  - get_cvp_info
  - add_user
  - delete_user
  - get_inventory
  - add_devices_to_inventory
  - add_container
  - move_device_to_container
  - add_configlet
  - add_configlet_builder
  - apply_configlets_to_container
  - apply_configlets_to_device
  - validate_configlets_for_device
  - add_image
  - save_image_bundle
  - pply_image_to_device

    … and more

- **cvp_client.py**
  - _login
  - logoff
  - get
  - post
  - delete

    … and more

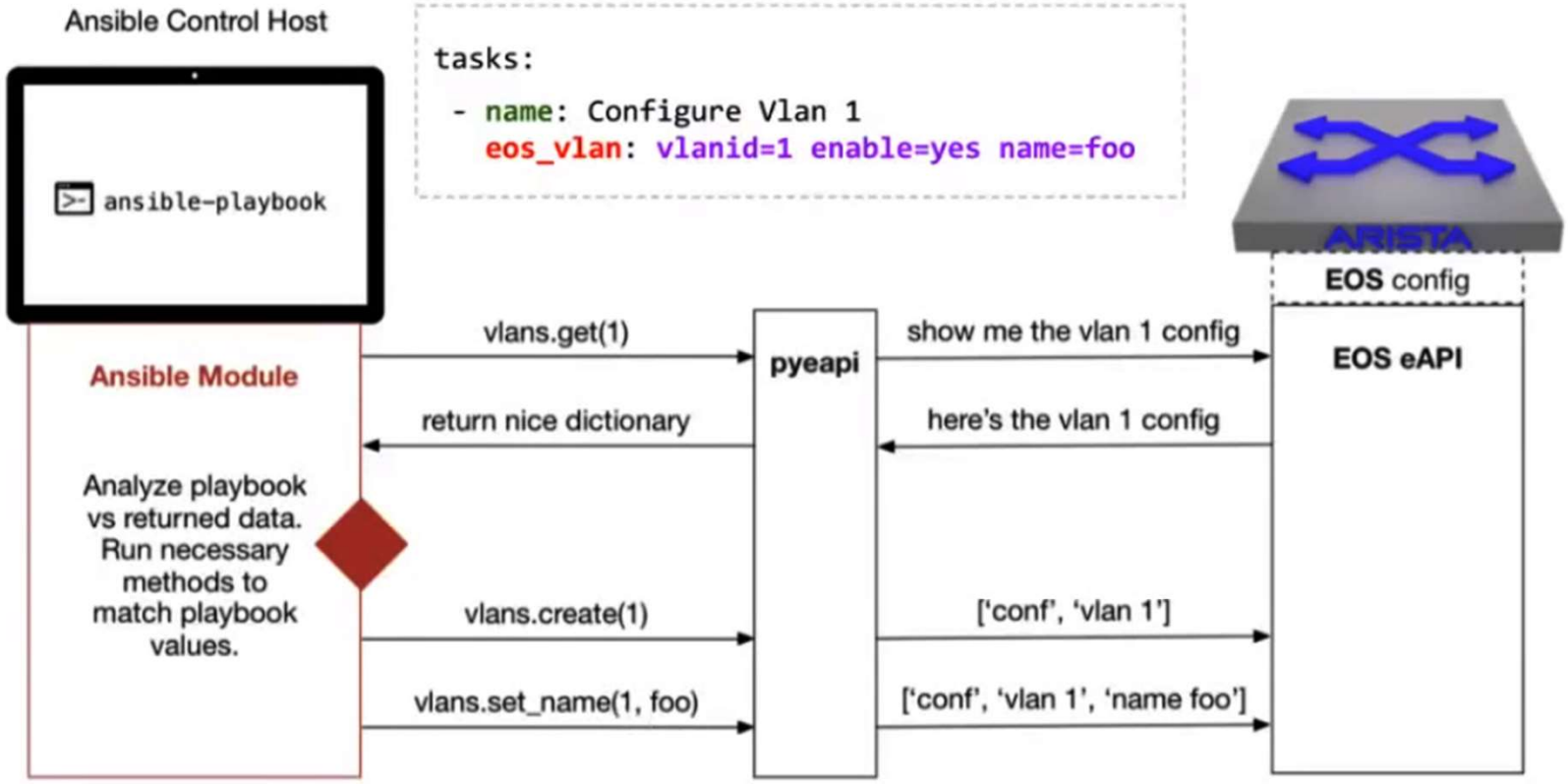
- **cvp_client_error.py**
  - CvpClientError
  - CvpApiError
  - CvpLoginError
  - CvpRequestError
  - CvpSessionLogOutError
  -

Pyeapi

A collection of modules to facilitate connection to Arista devices using python scripts

- **load_config.py :**
- **connect.py**
- **connect_to.py**
- **config_for.py**
- **vlans.py**
- **ospf.py**
- **stp.py**
- **user.py**
- **… and more**

How does it work?



Credit to Philip Dileo @Arista

Connection Methods

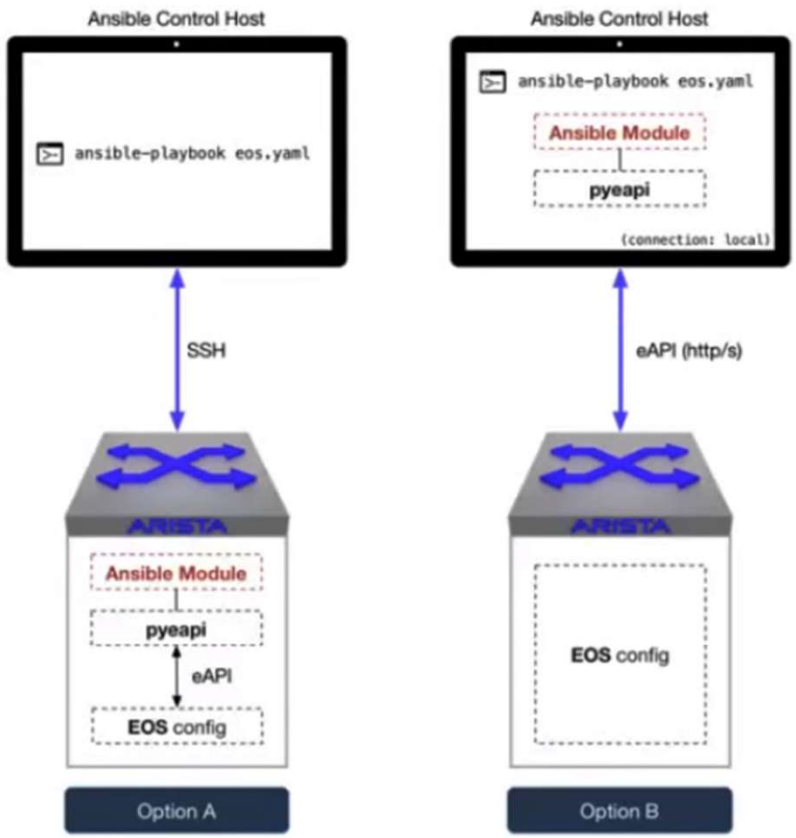
Option A - SSH

Requirements:

- Password-less SSH associations
- **pyeapi installed** on switch (you can do this with Ansible)
- eAPI enabled
- bash user

Notes:

- You can use http_local or unix sockets on >4.14.5F
- Technically more secure



Option B - eAPI

Requirements:

- **pyeapi installed** on Ansible Control Host
- eAPI enabled

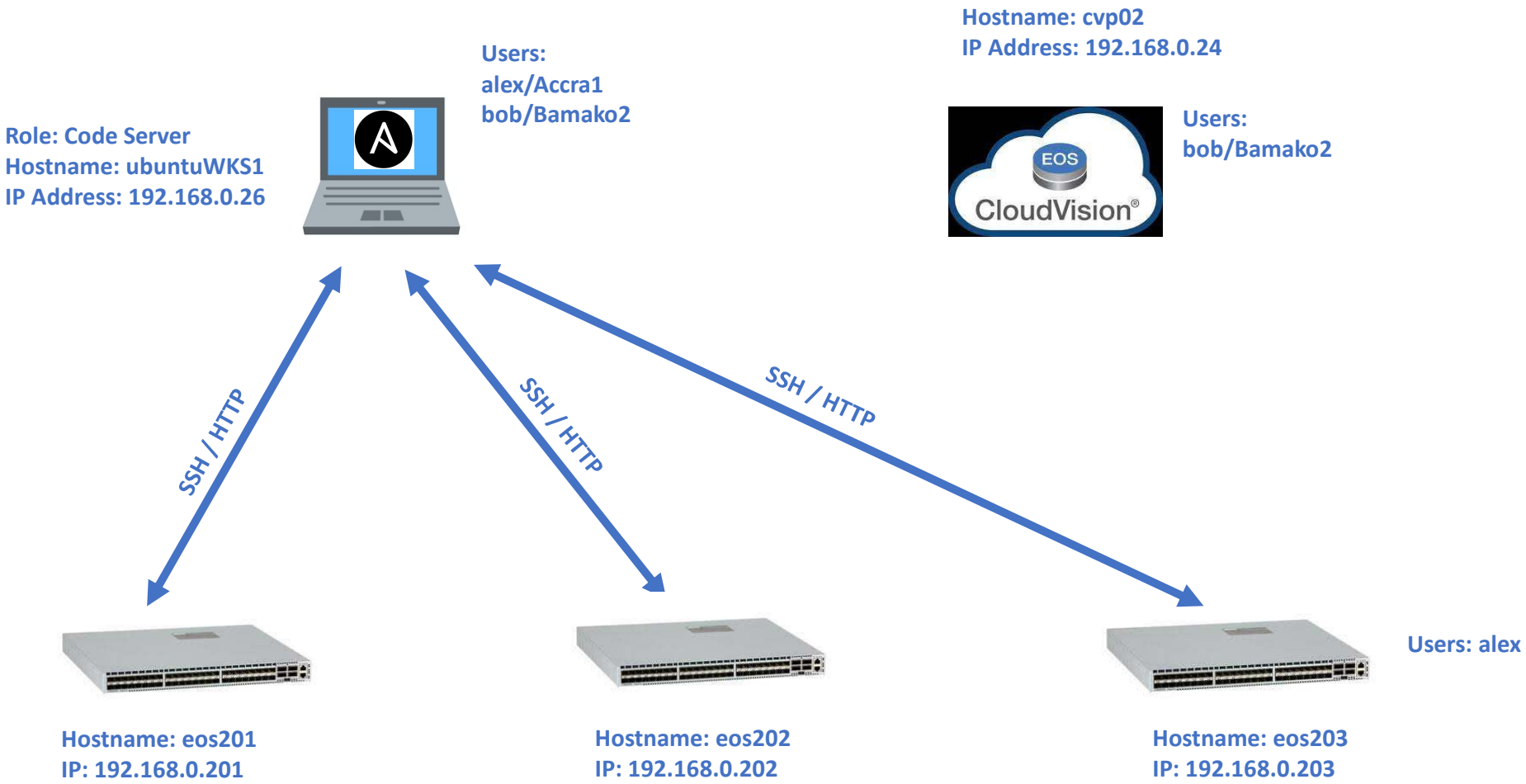
Notes:

- Simplicity but potentially less secure. Need to store eapi credentials in cleartext.

Credit to Philip Dileo @Arista

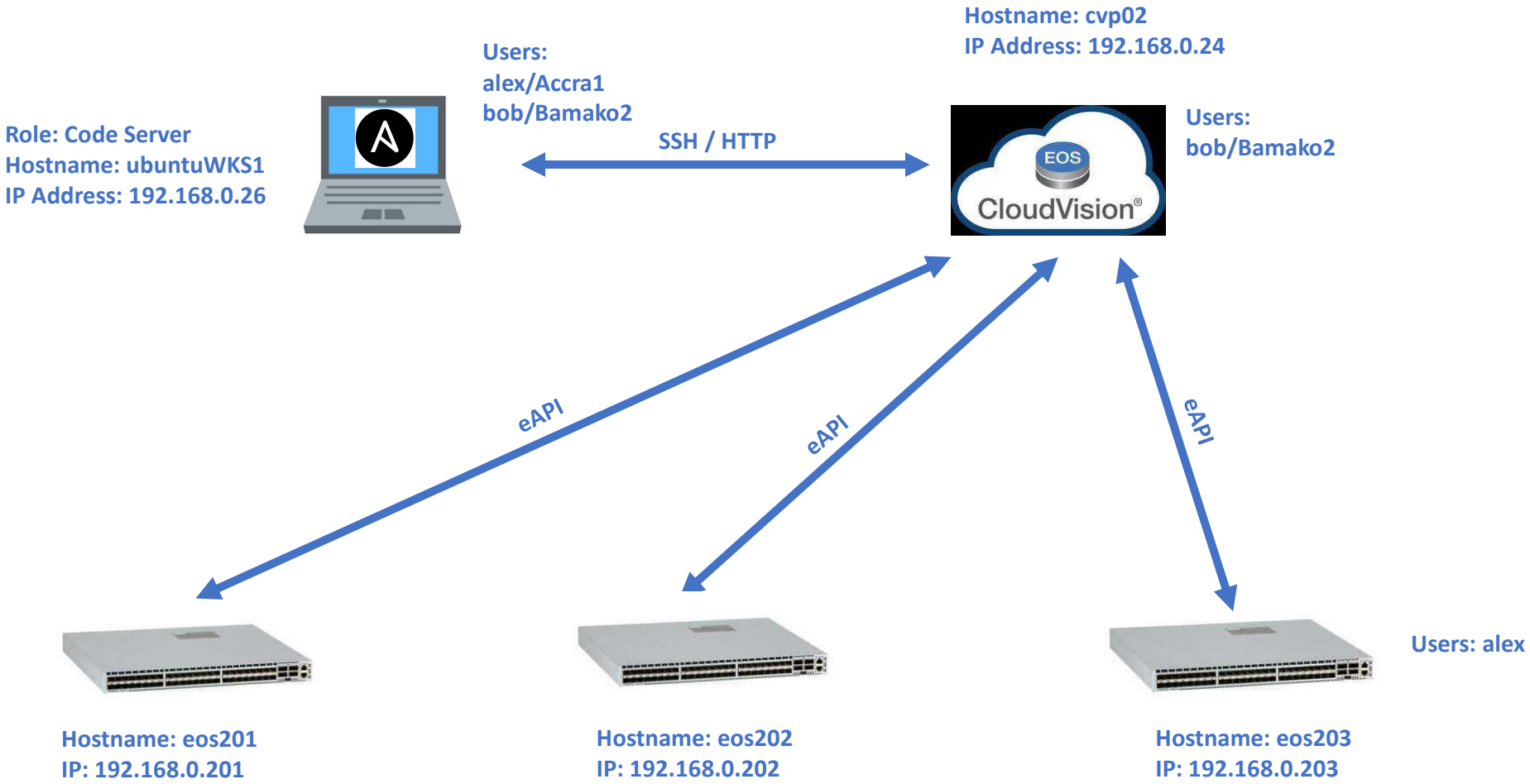
Ansible Example – Using Arista modules via eAPI

Demo



Ansible Example – Using Arista cvp modules via CVP

Demo



Ansible Roles

The grouping of multiple tasks into one container to perform automation in an efficient manner.

Ansible – Galaxy

- **Install roles**
  ~$ cd /etc/ansible/roles
  ~$ sudo ansible-galaxy init rolename
  ~$ cd rolename
  ~S ls

- **Folfers and files created**
  - defaults - files - handlers - meta
  - README.md - tasks - templates - test
  -vars

Ansible Directory Structure

- **Directory Structure**

  ~$ tree /etc/ansible/roles/rolename

  Roles/DC1

  README.md

  default

  main.yml

  handlers

  main.yml

  meta

  main.yml

  tasks

  main.yml

  vars

  main.yml

Optimizing Playbook – Ansible File System Layout

- Ansible Root Repository
  - ansible.cfg
  - inventory
  - site.yml
  - group_var
    - host_var
      - host1.yml
  - roles
    - role_name
      - task
        - main.yml
      - handler
        - main.yml
      - templates
        - template_filename.j2
      - file
        - filename.txt

Arista Validated Design (AVD)

- **arista.avd.eos_designs** - Opinionated Data model to assist with the deployment of Arista Validated Designs.
- **arista.avd.eos_cli_config_gen** - Generate Arista EOS cli syntax and device documentation.
- **arista.avd.eos_config_deploy_cvp** - Deploys intended configuration via CloudVision.
- **arista.avd.eos_config_deploy_eapi** - Deploys intended configuration via eAPI.
- **arista.avd.cvp_configlet_upload** - Uploads configlets from a local folder to CloudVision Server.
- **arista.avd.eos_validate_state** - Validate operational states of Arista EOS devices.
- **arista.avd.eos_snapshot** - Collect commands on EOS devices and generate reports.
- **arista.avd.dhcp_provisioner** - Configure an ISC-DHCP server to provide ZTP services and Cloudvision registration.

https://www.avd.sh

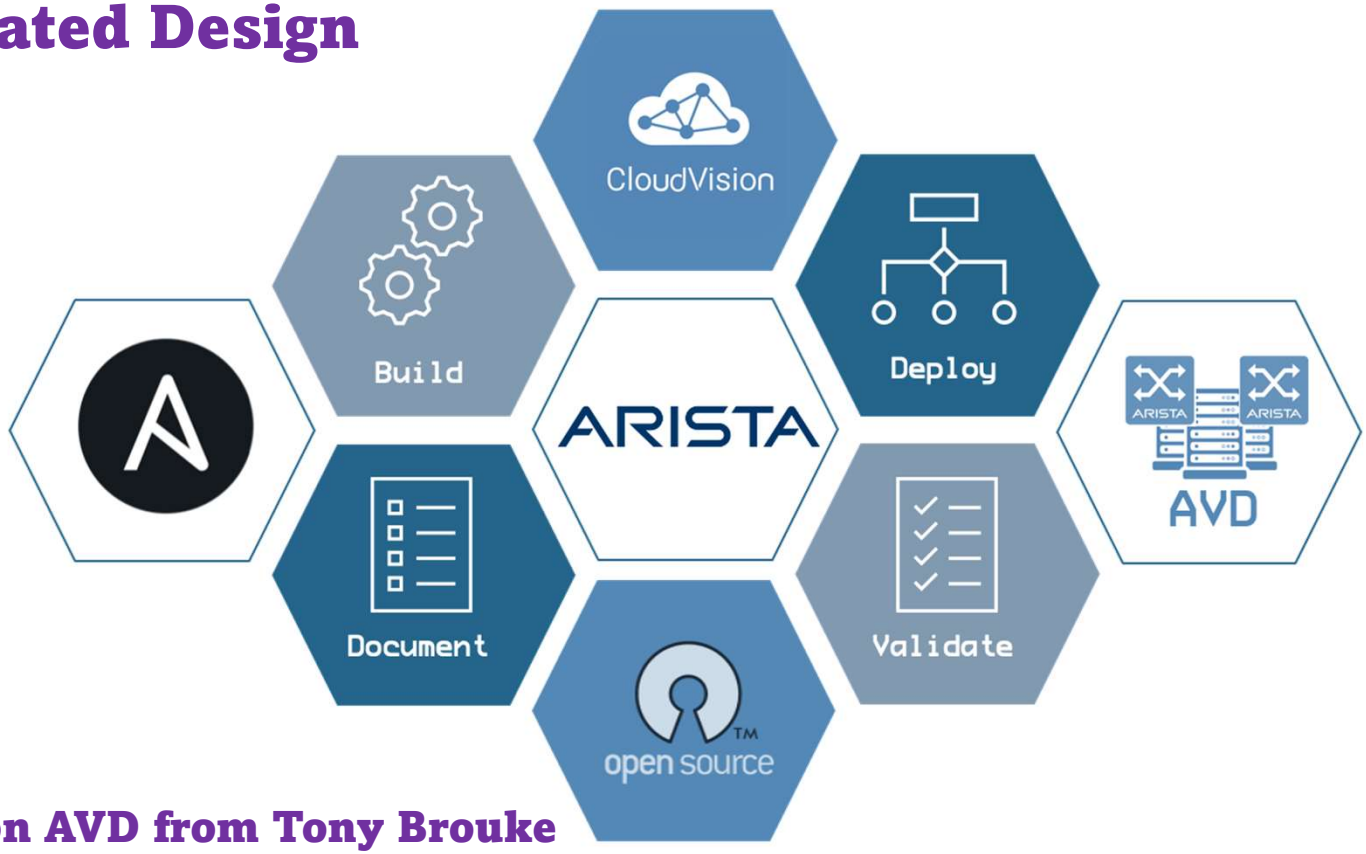
Ansible Modules - Arista AVD Roles

~/.ansible/collections/ansible_collections/arista/avd/roles

- **eos_designs**
- **eos_cli_config_gen**
- **eos_config_deploy_cvp**
- **eos_config_deploy_eapi**
- **cvp_configlet_upload**
- **eos_validate_state**
- **eos_snapshot**
- **dhcp_provisioner**

Arista Validated Design

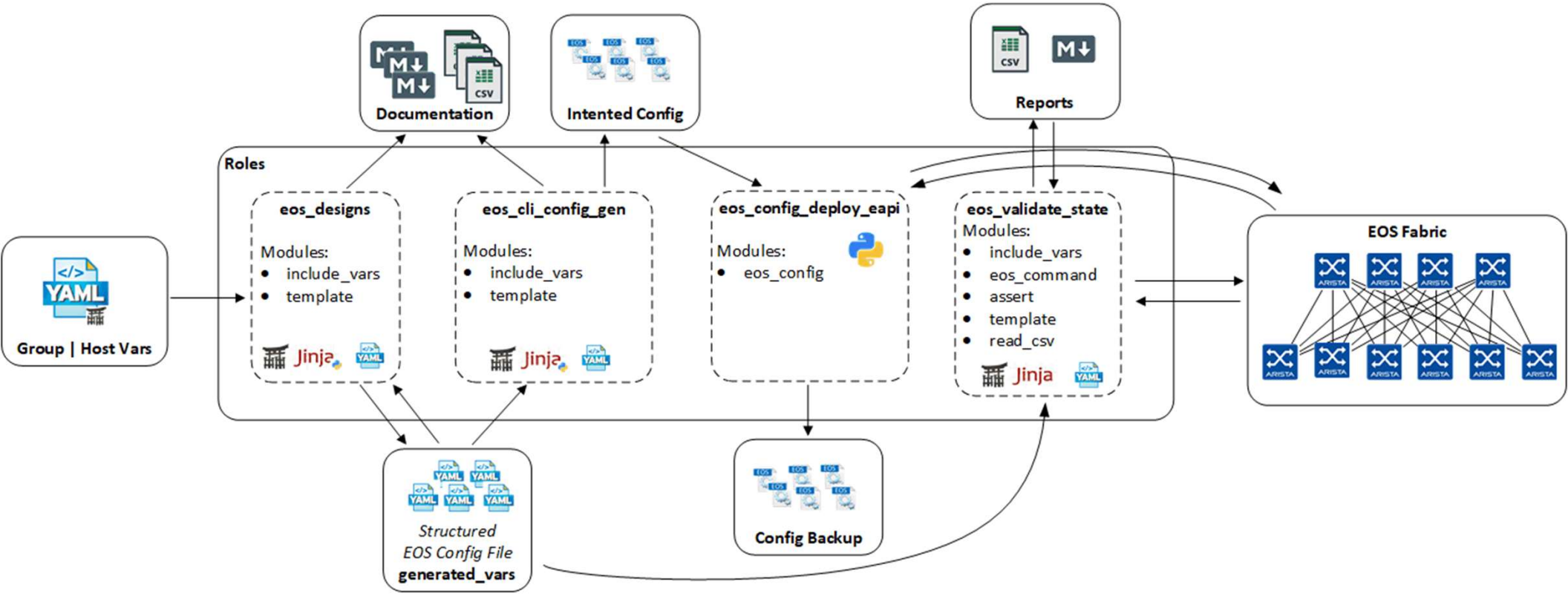
Avd.sh



Youtube Videos on AVD from Tony Brouke

https://www.youtube.com/watch?v=KTxrmCyJZ8w
https://www.youtube.com/watch?v=XAjnEo2HHoo
https://www.youtube.com/watch?v=YY0vlvsS12c

AVD eos_design direct using eAPI



AVD eos_design via CVP