**Comparison tool**

The “comparison tool” basically provides us an easy and handy method of conducting comparison between two components (folders/files), which can further have a list of various directories or different file formats. For each component ansible is listed with corresponding nodes details (inventory) and their tasks. For comparison of any components (across the env or within same env), corresponding tag need to execute along with the env name and then push the config into a git-repo to generate the report.

The tool basically works it two phases. In the first phase, a config dump is generating at the base path specified while execution, this config dump is taken node wise i.e., files from one node are clubbed together inside a directory that is named after the node. The config dump can be generated in various scenarios/options available in the tool. In the second phase this dump is then uploaded to a git-repo one after the other thus showing the differences between both the dumps.

* + - * **Objective:**

The tool is basically designed for the SI team of the project for managing and verifying the configurations of different components across different labs/env. It is built to save time & improve resource utilization by providing quick and handy config-diff reports, highlighting the differences, which would otherwise consume hours of manual work.

It comes in handy primarily in the config verification, where a user-friendly report showing the differences or the delta between primary and the secondary config dump is generated (using git commit).

The tool is extensively used in the deployment config verifications and troubleshooting of various issues in Test-labs/Prod.

* + - * **Options available in tool:**
* Fetch files & compare
* Collect md5sum & compare
* Collect rpm versions for ECE
* Collect SDG/MSDP/BNE components version & compare
  + - * **Commands:**

ansible-playbook capture.yml -i inventory*/<env.yaml>*  -e "base\_path*=<tmp-config-path>"* --tags=*<tag>* -D

***<env.yaml>*** *: Refers to the env user is trying to run the commands for, env specific yaml files are there in “inventory” directory.*

***<tmp-config-path>*** *: Refers to the path were user wants to collect the dump.*

***<tag>*** *: Refers to the component for which the user wants to execute the tool. All tags are available in “capture.yml”*

*Note: these are elaborated further in doc.*

* + - * **Tool’s directory structure:**

1. inventory
2. playbooks
3. roles
   1. capture-archive-hash
   2. fetch-files
   3. fetch-version
   4. fetch-version-bne
   5. rpm-check
4. ansible.cfg
5. capture.yml
6. README.md

* + - * **Accessing the outputs:**

The primary output is the corresponding commit made in git which shows the delta or the differences. (For reference, please refer to Appendix D.)

Note: one should primarily give a unique and relevant commit message which can be referred later.

* **Working of the tool:**

**Phase1: Running the ansible for Config Dump**

In the first phase the user is supposed to generate or rather collect the required dump by running the ansible script. Once the ansible script is executed for the specified *env* the dump relevant to the *tag* specified in the command will be collected node wise in the *tmp-config-path* in the localhost.

**Phase2: Push to Git Repo**

In the second phase once the config dump is collected in the above step, it is then pushed into a git repo one after the other in a new blank directory. All the push in this process have their specific commit message that specify their content.

There are different options/tasks that can be performed in the tool. End to end execution templates are specified below:

1. Create backup of single/multiple component*:*

We are just taking the config dump and storing it into our git repo for future references

Phase1: Running the ansible for Config Dump

ansible-playbook capture.yml -i inventory*/<env.yaml>* -e "base\_path*=<tmp-config-path>"* --tags=*<tag>* -D

*For <env.yaml> refer Appendix A*

*For <tmp-config-path> refer Appendix B*

*For <tag> refer Appendix C { All the tags can be used }*

Phase2: Push to Git Repo

Refer to example in Appendix D

1. Fetch files & md5sum to compare (Comparison between two components):

Phase1: Running the ansible for Config Dump :-

ansible-playbook capture.yml -i inventory*/<env.yaml>* -e "base\_path*=<tmp-config-path>"* --tags=*<tag>* -D

*For <env.yaml> refer Appendix A*

*For <tmp-config-path> refer Appendix B*

*For <tag> refer Appendix C {ECE, BNE, SDG/MSDP }*

Phase2: Push to Git Repo

Refer to example in Appendix D

1. Collect rpm versions for ECE:

Phase1: Running the ansible for Config Dump

ansible-playbook capture.yml -i inventory*/<env.yaml>* -e "base\_path*=<tmp-config-path>"* --tags=*<tag>* -D

*For <env.yaml> refer Appendix A*

*For <tmp-config-path> refer Appendix B*

*For <tag> refer Appendix C { ECE-Traffic-RPM }*

Phase2: Push to Git Repo

Refer to example in Appendix D

1. Collect SDG/MSDP/BNE components version & compare

Phase1: Running the ansible for Config Dump

ansible-playbook capture.yml -i inventory*/<env.yaml>* -e "base\_path*=<tmp-config-path>"* --tags=*<tag>* -D

*For <env.yaml> refer Appendix A*

*For <tmp-config-path> refer Appendix B*

*For <tag> refer Appendix C {BNE Version, SDG/MSDP-Version }*

Phase2: Push to Git Repo

Refer to example in Appendix D

* **TOOLS DIRECTORY STRUCTURE:**

1. inventory : this directory contains all the relevant IPs specific to an env, under a yam filel <env.yaml>
2. playbooks: this directory contains all the yaml files specific to each component
3. roles: this directory contains the roles/task that are performed by the tool, which are listed below:-
   1. capture-archive-hash
   2. fetch-files
   3. fetch-version
   4. fetch-version-bne
   5. rpm-check
4. ansible.cfg: it is a configuration file that contains all the relevant ansible configurations.
5. capture.yml: this file contains all the tags that are to be used in the tool
6. README.md: this file contains quick data on how to run the ansible to collect config dump.

* **PROCEDURE OF ADDING NEW ENV, NODE OR COMPONENT TO THE TOOL:**
* ADDING NEW ENV:

For addition of a new env to the existing tool, a new yaml file ( <env\_name.yaml> ) is to be added in the inventory directory. This new file should have all the relevant IP details and node groupings like the existing files in that directory.

* ADDING NEW NODE:

For addition of new node to an existing env, the corresponding IP details are to be added in the relevant yaml file ( <env\_name.yaml> ). Also the node name is to be mentioned in the relevant node groupings, of the same yaml file.

* ADDING NEW COMPONENT:

For any new component to be added to the existing tool, below steps are to be followed:

1. Add a yaml file in the playbooks directory that should contain all the tasks that are to be executed for that component. The name of the yaml file should be proper and unique. It will also contain the detail of a file (in group\_vars) that would contain all the variables for that component.
2. Create the var file mentioned in above step in “playbooks/greoup\_vars” directory. This file will have the values for the required variables for this new component (similar to the existing files). One should properly define all the values , relevant to the new component.
3. In this step a tag is assigned for this new component in capture.yml file (similar to the existing once). the tag should be unique and relevant enough for future users to understand.
4. Also optionally you can add the details into README.md file.

*Appendix A: [ For <env.yaml> ]*

<env.yaml> : this variable refers to the yaml file for the env/site for which the user is trying to run the commands for. These files can be easily found inside the inventory directory of the script.

The yaml file serves as an inventory for that env containing all the relevant IP details of the nodes.

*Appendix B: [ FOR <tmp-config-path> ]*

<tmp-config-path>: this variable is used to specify the path where the config dump can be collected. Preferred location is the tmp directory.

*Appendix C: [ For <tag> ]*

<tag> : all the required tags are specified in the capture.yml file. User can just select the relevant tags and use them in separate or same command (separated by “,”)

the tags that are currently being used are listed below:

ECE:-

* exposure-traffic
* payment-stc
* bne-traffic
* common-service-traffic
* pic-components-cluster
* pic-components-opt
* Exposure-BM
* Foundation-BM
* ESA
* policy-api

SDG/MSDP:

* 3pi
* provision
* sdg-3ge-conf
* SP-API
* customer-care
* runtime
* manager
* orchestration
* spending-bucket

BNE:

* bne-admin
* bne\_mo
* message\_carousel\_kafka
* message\_carousel\_immediate
* bne\_bulk\_notification
* bne\_file\_notification
* bne\_file\_report
* message\_delivery
* bne\_partner\_notifications
* bne\_event\_handler\_2
* bne\_dr\_handler
* bne\_event\_handler\_1
* message\_carousel\_scheduled

ECE-Traffic-RPM:

* ece-traffic-rpms

SDG/MSDP-Version:

* sdg-version

BNE-Version:

* bne-version

*Appendix D: [ Phase2: Push to Git Repo ]*

*Below is an example:*

[ansible@SGRERCA1 ansible]$ cd /home/ansible/gitlab/config-management-repo2

[ansible@SGRERCA1 config-management-repo2]$ ll

total 0

drwxrwxr-x. 2 ansible ansible 19 Sep 7 03:03 MY\_LAB\_123

drwxrwxr-x. 2 ansible ansible 19 Sep 7 03:03 MY\_LAB\_123-compare

[ansible@SGRERCA1 config-management-repo2]$ cd MY\_LAB\_123/

*COPY THE CONFIG FILES*

[ansible@SGRERCA1 MY\_LAB\_123]$ cp -rp /tmp/test\_config/\* .

[ansible@SGRERCA1 MY\_LAB\_123]$ ll

total 0

drwxrwxr-x. 3 ansible ansible 45 Sep 8 00:23 NODE1

drwxrwxr-x. 3 ansible ansible 45 Sep 8 00:23 NODE2

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ cd ..

*GIT ADD & COMMIT*

[ansible@SGRERCA1 config-management-repo2]$ git add MY\_LAB\_123

[ansible@SGRERCA1 config-management-repo2]$ git commit -m "adding config files"

[ansible@SGRERCA1 MY\_LAB\_123]$ cd MY\_LAB\_123-compare/

*REMOVE ANY PREVIOUS FILE FROM COMPARE DIRECTORY*

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ ll

total 0

-rw-rw-r--. 1 ansible ansible 0 Sep 7 03:03 a.txt

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ rm -rf a.txt

*COPY THE PRIMARY CONFIG FILES*

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ cp -rp ../MY\_LAB\_123/NODE1 .

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ ll

total 0

drwxrwxr-x. 3 ansible ansible 45 Sep 8 00:23 NODE1

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ cd ..

*GIT ADD & COMMIT*

[ansible@SGRERCA1 config-management-repo2]$ git add MY\_LAB\_123-compare

[ansible@SGRERCA1 config-management-repo2]$ git commit -m "adding 1st config"

[ansible@SGRERCA1 config-management-repo2]$ cd MY\_LAB\_123-compare/

*REMOVE THE PRIMARY CONFIG FILES*

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ ll

total 0

drwxrwxr-x. 3 ansible ansible 45 Sep 8 00:23 NODE1

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ rm -rf NODE1/

*COPY SECONDARY CONFIG FILES*

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ cp -rp ../MY\_LAB\_123/NODE2 .

*NAME OF PRIMARY AND SECONDARY CONFIG DIRCTORY SHOULD BE SAME, HENCE RENAMING*

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ ll

total 0

drwxrwxr-x. 3 ansible ansible 45 Sep 8 00:23 NODE2

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ mv NODE2/ NODE1

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ ll

total 0

drwxrwxr-x. 3 ansible ansible 45 Sep 8 00:23 NODE1

[ansible@SGRERCA1 MY\_LAB\_123-compare]$ cd ..

[ansible@SGRERCA1 config-management-repo2]$

*GIT ADD, COMMIT & PUSH*

[ansible@SGRERCA1 config-management-repo2]$ git add MY\_LAB\_123-compare

[ansible@SGRERCA1 config-management-repo2]$ git commit -m "config diff report NODE1 vs NODE2 "

[ansible@SGRERCA1 config-management-repo2]$ git push

Counting objects: 547, done.

Delta compression using up to 2 threads.

Compressing objects: 100% (395/395), done.

Writing objects: 100% (545/545), 3.17 MiB | 0 bytes/s, done.

Total 545 (delta 159), reused 413 (delta 125)

remote: Resolving deltas: 100% (159/159), completed with 1 local object.

da21668..64eb3c5 main -> main

[ansible@SGRERCA1 config-management-repo2]$