# Using Regular Expressions in Ansible

*By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years*

**Session - 27 Agenda:**
1. Using Regular Expressions in Ansible

**Regular Expressions:**
Regexps are acronyms for regular expressions. Regular expressions are special characters or sets of characters that help us to search for data and match the complex pattern. Regexps are most commonly used with the Linux commands like grep, sed, tr & vi.

There are three different versions of regular expression syntax:
**BRE:** Basic Regular Expressions
**ERE:** Extended Regular Expressions
**PRCE:** Perl Regular Expressions

Depending on the tool being used, one or more of these syntaxes can be used. For example, the grep tool has the -E option to force a string to be read as ERE while -G forces BRE and -P forces PRCE.
*(Please follow the Regular Expressions session of our RHCSA playlist first to go in the details.)*

The following are some basic regular expressions:

| S. No. | Symbol | Description |
|---|---|---|
| 1. | . | It is called a wild card character; it matches any one character other than the new line. |
| 2. | ^ | It matches the start of the string. |
| 3. | $ | It matches the end of the string. |
| 4. | * | It matches up to zero or more occurrences i.e., any number of times of the character of the string. |
| 5. | \ | It is used for escape following character. |
| 6. | () | It is used to match or search for a set of regular expressions. |
| 7. | ? | It matches exactly one character in the string or stream. |

**GREP (Global Regular Expression Print):**
grep is a popular Linux tool to search for lines that match a certain pattern.

**1. Print The Lines Matching a Pattern:**
Below are some examples of the simplest regular expressions. This is the contents of the test file. This file contains three lines (or three newline characters).
**$ vim names.txt**
**Tania**
**Laura**
**Valentina**

When grepping for a single character, only the lines containing that character are returned.
**$ grep u names.txt**
**Laura**

**$ grep e names.txt**
**Valentina**

**$ grep i names.txt**
**Tania**
**Valentina**

The pattern matching in this example should be very straightforward; if the given character occurs on a line, then grep will return that line.

# Using Regular Expressions in Ansible

*By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years*

**2. Concatenating Characters:**

Two concatenated characters will have to be concatenated in the same way to have a match. This example demonstrates that 'ia' will match Tania but not Valentina and in will match Valentina but not Tania.

**$ grep a names.txt**
**Tania**
**Laura**
**Valentina**

**$ grep ia names.txt**
**Tania**

**$ grep in names.txt**
**Valentina**

**3. One or The Other:**

PRCE and ERE both use the pipe symbol to signify OR. In this example we grep for lines containing the letter i or the letter a.

**$ vim list.txt**
**Tania**
**Laura**

**$ grep -E 'i|a' list.txt**
**Tania**
**Laura**

Note that we use the -E switch of grep to force interpretation of our string as an ERE.

We need to escape the pipe symbol in a BRE to get the same logical OR.

**$ grep -G 'i|a' list.txt**

**$ grep -G 'i\|a' list.txt**
**Tania**
**Laura**

**4. One or More:**

The * signifies zero, one or more occurrences of the previous and the + signifies one or more of the previous.

**$ cat list2.txt**
**ll**
**lol**
**lool**
**loool**

**$ grep -E 'o*' list2.txt**
**ll**
**lol**
**lool**
**loool**

**$ grep -E 'o+' list2.txt**
**lol**
**lool**
**loool**

# Using Regular Expressions in Ansible

*By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years*

**5. Match The End of a String:**
**$ vim names.txt**
**Tania**
**Laura**
**Valentina**
**Fleur**
**Floor**

The two examples below show how to use the dollar character to match the end of a string.
**$ grep a$ names.txt**
**Tania**
**Laura**
**Valentina**

**$ grep r$ names.txt**
**Fleur**
**Floor**

**6. Match The Start of a String:**
The caret character (^) will match a string at the start (or the beginning) of a line. Given the same file as above, here are two examples.
**$ grep ^Val names.txt**
**Valentina**

**$ grep ^F names.txt**
**Fleur**
**Floor**

Both the dollar sign and the little hat are called anchors in a regex.

**7. Separating Words:**
Regular expressions use a \b sequence to reference a word separator. Take for example this file:
**$ vim text**
**The governer is governing.**
**The winter is over.**
**Can you get over there?**

Simply grepping for over will give too many results.
**$ grep over text**
**The governer is governing.**
**The winter is over.**
**Can you get over there?**

Surrounding the searched word with spaces is not a good solution (because other characters can be word separators). This screenshot below shows how to use \b to find only the searched word:
**$ grep '\bover\b' text**
**The winter is over.**
**Can you get over there?**

Note that grep also has a -w option to grep for words.
**$ grep -w over text**
**The winter is over.**
**Can you get over there?**

# Using Regular Expressions in Ansible

*By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years*

**Using Regular Expressions in Ansible:**

We can use regular expressions in Ansible also as per the requirement of the automation task e.g. to make changes in any file. Let's understand it with an example:

Suppose we want to make SELINUX mode as enforcing/permissive in the /etc/selinux/config file on a managed node using Ansible.

**$ cat /etc/selinux/config**

**$ sestatus**

Let's create an ansible playbook for making SELINUX mode as permissive.

**$ vim selinux.yml**

```
---
- name: Change SELINUX mode in the configuration file.
  hosts: node1
  become: true
  tasks:
     - name: Changing the SELINUX mode in the /etc/selinux/config file.
       lineinfile:
            path: /etc/selinux/config
            regexp: '^SELINUX='
            line: 'SELINUX=permissive'
...
```

Check the playbook for the syntax errors.

**$ ansible-playbook selinux.yml --syntax-check**

Execute the playbook to make selinux mode as permissive on node1.

**$ ansible-playbook selinux.yml**

Verify the same using ansible ad-hoc command.

**$ ansible node1 -m command -a 'sestatus'**

**$ ansible node1 -m command -a 'cat /etc/selinux/config'**

**$ ansible node1 -m command -a 'grep SELINUX= /etc/selinux/config'**

Suppose now we need to change SELINUX mode as enforcing on node1. For this job either we need to make change in the playbook (write there enforcing instead of permissive) or we can use variables.

**$ vim selinux2.yml**

```
---
- name: Change SELINUX mode in the configuration file.
  hosts: node1
  become: true
  vars:
     status: enforcing
  tasks:
     - name: Changing the SELINUX mode in the /etc/selinux/config file.
       lineinfile:
            path: /etc/selinux/config
            regexp: '^SELINUX='
            line: 'SELINUX={{ status }}'
...
```

Check the playbook for the syntax errors.

**$ ansible-playbook selinux2.yml --syntax-check**

# Using Regular Expressions in Ansible

*By: Er. Vikas Nehra (M. Tech, B. Tech), Experience: 15 + Years*

Execute the playbook to make selinux mode as permissive on node1.
**$ ansible-playbook selinux2.yml**

Verify the same using ansible ad-hoc command.
**$ ansible node1 -m command -a 'sestatus'**

**$ ansible node1 -m command -a 'cat /etc/selinux/config'**

**$ ansible node1 -m command -a 'grep SELINUX= /etc/selinux/config'**

Variables in the playbook provides you the flexibility to perform the desired action without making changes in the playbook. Suppose if again we need to make SELINUX mode as permissive on node1, we can directly do it from the command while executing the same playbook.
**$ ansible-playbook selinux2.yml -e status=permissive**

Verify the same using ansible ad-hoc command.
**$ ansible node1 -m command -a 'sestatus'**

**$ ansible node1 -m command -a 'cat /etc/selinux/config'**

**$ ansible node1 -m command -a 'grep SELINUX= /etc/selinux/config'**

---

**Thank You**