YAML - SEQUENCES

Yaml Breakdown

- Mapping
- **Sequences**
- Scalars
- Structures
- Comments
- Tags
- Anchors

Sequences

- Creating lists and arrays in YAML.
- Nesting sequences.
- Practical applications.

What Are Sequences In Yaml?

- A sequence is a type of data structure that allows you to represent an ordered collection of items.
- Sequences are similar to lists or arrays in other programming languages.
- The order of elements in a sequence is preserved, and each item in the sequence can be of any data type, including other sequences or mappings.
- Sequences are represented using dashes (-) followed by a space to denote each item in the list.

Example:

```
---
- item1
- item2
- item3
```

Creating Lists And Arrays

- A sequence is an ordered collection of items, and each item is denoted by a dash (-) followed by a space.
- Lists and arrays are represented using sequences.
- Items can be of any data type and can include other sequences or mappings.

Examples:

```yaml
---
fruits:
  - apple
  - banana
  - orange
```

Creating List

```yaml
---
matrix:
  - - 1
    - 2
    - 3
  - - 4
    - 5
    - 6
  - - 7
    - 8
    - 9
```

Creating Array

```yaml
---
mixed_list:
  - 42
  - "hello"
  - true
  - [1, 2, 3]
```

Creating Mixed Type

Nested Sequences

- A nested sequence refers to a sequence (a list of items) that contains other sequences as its elements.
- This allows for a hierarchical and multi-dimensional data structure.
- Each level of nesting indicates a new level of the data hierarchy.

Example:

```
---
nested_sequence:
  - - item11
    - item12
    - item13
  - - item21
    - item22
  - - item31
    - - subitem311
```

Use-cases Of Nested Sequence

- Representing multi-level data hierarchies, such as organizational structures, directories, or matrices.
- Configuring complex systems where components have subcomponents or configurations at different levels.
- Storing multi-dimensional data like a table, grid, or matrix.

Applications of Sequences Are:

- Defining host group and group variables.
- Defining task in playbook.
- Defining role dependencies.
- Using looping constructs.
- Defining role variables.

Examples:


```yaml
---
all:
  children:
    web_servers:
      hosts:
        - web1
        - web2
    db_servers:
      hosts:
        - db1
        - db2
  vars:
    some_var: value
```

Defining Host Group

```yaml
---
tasks:
  - name: Install packages
    yum:
      name:
        - package1
        - package2
    state: present
  - name: Configure service
    template:
      src: service.conf.j2
      dest: /etc/service/config.conf
```

Defining Task

```yaml
---
roles:
  - role: my_role
    vars:
      my_var: value
      my_list:
        - item1
        - item2
```

Defining Role Variables