This project demonstrates how to create conditional workflows in Argo using simple and complex conditions, parallel processing, and actions based on random events (coin flips). It helps trainees understand workflow logic and how to achieve specific actions based on conditions and events within a workflow.

This YAML configuration defines an Argo Workflow that simulates flipping a coin and performing different actions based on the result. Let's break down what this project does and how it achieves its functionality in a simple way for trainees:

1. Coin Flipping Workflow:

- The workflow is named `coinflip` and includes multiple templates for different actions based on the result of coin flips.

2. Templates:

- `flip-coin` Template: This template uses a Python script to simulate flipping a coin and generates a result of either "heads" or "tails" randomly.

- `heads` and `tails` Templates: These templates simply echo a message indicating whether the result was "heads" or "tails."

- `heads-tails-or-twice-tails` Template: This template echoes a message based on a more complex condition. It checks if the first flip was "heads" and the second was "tails," or if both flips were "tails."

3. Workflow Logic:

- The workflow starts by flipping a coin (`flip-coin` template).

- It then evaluates the result in parallel:
  - If the result is "heads," it executes the `heads` template.
  - If the result is "tails," it executes the `tails` template.

- After the first flip, it flips the coin again (`flip-again` template).

- It then evaluates a more complex condition (`complex-condition` template):
  - If the first flip was "heads" and the second was "tails," or if both flips were "tails," it executes the `heads-tails-or-twice-tails` template.

- Additionally, it checks if the result of the second flip matches a regular expression (`heads-regex` and `tails-regex` templates) and executes the corresponding template based on the match.

4. Execution:

- When the workflow runs, it goes through each step sequentially, starting with flipping the coin, evaluating the results, and executing the appropriate templates based on the conditions.

- Each template is a separate action or command that is executed based on the logic defined in the workflow.

Overall, this project demonstrates how to create conditional workflows in Argo using simple and complex conditions, parallel processing, and actions based on random events (coin flips). It helps trainees understand workflow logic and how to achieve specific actions based on conditions and events within a workflow.

In this example we flip 2 times a coin. First time we

use the simple conditionals syntax. The second time we use

regex and a complex condition with logical AND and OR.

We also use of the parenthesis for defining the priority.

```
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: coinflip-
  namespace: argo
spec:
  entrypoint: coinflip
  serviceAccountName: training
  templates:
  - name: coinflip
    steps:
    # flip a coin
    - - name: flip-coin
        template: flip-coin
    # evaluate the result in parallel
    - - name: heads
```

```
      template: heads                    # call heads template if "heads"
      when: "{{steps.flip-coin.outputs.result}} == heads"
    - name: tails
      template: tails                    # call tails template if "tails"
      when: "{{steps.flip-coin.outputs.result}} == tails"
  - - name: flip-again
      template: flip-coin
  - - name: complex-condition
      template: heads-tails-or-twice-tails
      # call heads template if first flip was "heads" and second was "tails" OR both were "tails"
      when: >-
        ( {{steps.flip-coin.outputs.result}} == heads &&
          {{steps.flip-again.outputs.result}} == tails
        ) ||
        ( {{steps.flip-coin.outputs.result}} == tails &&
          {{steps.flip-again.outputs.result}} == tails )
    - name: heads-regex
      template: heads                    # call heads template if ~ "hea"
      when: "{{steps.flip-again.outputs.result}} =~ hea"
    - name: tails-regex
      template: tails                    # call heads template if ~ "tai"
      when: "{{steps.flip-again.outputs.result}} =~ tai"

# Return heads or tails based on a random number
- name: flip-coin
  script:
    image: python:alpine3.6
    command: [python]
    source: |
      import random
      result = "heads" if random.randint(0,1) == 0 else "tails"
      print(result)

- name: heads
  container:
```

```
    image: alpine:3.6
    command: [sh, -c]
    args: ["echo \"it was heads\""]

  - name: tails
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was tails\""]

  - name: heads-tails-or-twice-tails
    container:
      image: alpine:3.6
      command: [sh, -c]
      args: ["echo \"it was heads the first flip and tails the second. Or it was two times tails.\""]
```

Recursive

```yaml
apiVersion: argoproj.io/v1alpha1
kind: Workflow
metadata:
  generateName: coinflip-recursive-
  namespace: argo
spec:
  entrypoint: coinflip
  serviceAccountName: training
  templates:
  - name: coinflip
    steps:
    # flip a coin
    - - name: flip-coin
        template: flip-coin
    # evaluate the result in parallel
    - - name: heads
        template: heads                 # call heads template if "heads"
        when: "{{steps.flip-coin.outputs.result}} == heads"
      - name: tails                     # keep flipping coins if "tails"
        template: coinflip
        when: "{{steps.flip-coin.outputs.result}} == tails"
```

```yaml
- name: flip-coin
  script:
    image: python:alpine3.6
    command: [python]
    source: |
      import random
      result = "heads" if random.randint(0,1) == 0 else "tails"
      print(result)

- name: heads
  container:
    image: alpine:3.6
    command: [sh, -c]
    args: ["echo \"it was heads\""]
```