Jenkins' Pipeline syntax

Internal reference: topics/03-2.md

Pipeline syntax

In this course we use the declarative syntax that is recommended for all new projects. The other options are a Groovy-based DSL and XML (created through the web interface). The declarative syntax was designed to make it as simple as possible to understand the pipeline, even by people who do not write code on a daily basis. This is why the syntax is limited only to the most important keywords.

Let's try an example and read the comments carefully:

```
pipeline {
     //Uses any available agent
     agent any
     //Triggers every 15 minutes     * see more
https://linuxhandbook.com/crontab/
     triggers { cron('H/15   * * * *') }
     //Stops if the execution takes more than 5 minutes
     options { timeout(time: 5) }
     //Asks for the Boolean input parameter before starting
     parameters {
          booleanParam(name: 'DEBUG_BUILD', defaultValue: true,
          description: 'Is it the debug build?')
     }
     stages {
          stage('Example') {
               //Sets Node-Backend as the NAME environment variable
               environment { NAME = 'Node-Backend' }
               when { expression { return params.DEBUG_BUILD } }
               steps {
                    echo Building $NAME
                    script {
                         def browsers = ['chrome', 'firefox']
                         for (int i = 0; i < browsers.size(); ++i) {
                              echo Testing the ${browsers[i]} browser.
                         }
                    }
               }
          }
     }
     post { always { echo 'I will always say Hello again!' } }
}
```

What are Sections?

Sections define the pipeline structure and usually contain one or more directives or steps. They are defined with the following keywords:

- `Stages`: This defines a series of one or more stage directives.
- `Steps`: This defines a series of one or more step instructions.
- `Post`: This defines a series of one or more step instructions that are run at the end of the pipeline build; they are marked with a condition (i.e. always, success, or failure) and are usually used to send notifications after the pipeline build.
- `Agent`: This specifies where the execution takes place and can define label to match the equally labeled agents, or docker to specify a container that is dynamically provisioned to provide an environment for the pipeline execution.

What are Directives?

Directives express the configuration of a pipeline or its parts:

- `Triggers`: This defines automated ways to trigger the pipeline and can use cron to set the time-based scheduling.
- `Options`: This specifies pipeline-specific options – for example, timeout (the maximum time of a pipeline run) or retry (the number of times the pipeline should be rerun after failure).
- `Environment`: This defines a set of key values used as environment variables during the build.
- `Parameters`: This defines a list of user-input parameters.
- `Stage`: This allows for the logical grouping of steps.
- `When`: This determines whether the stage should be executed, depending on the given condition.
- `Tools`: This defines the tools to install and put on PATH.
- `Input`: This allows us to prompt the input parameters.
- `Parallel`: This allows us to specify stages that are run in parallel.
- `Matrix`: This allows us to specify combinations of parameters for which the given stages run in parallel.

What are Steps?

Steps are the most fundamental part of the pipeline. They define the operations that are executed, so they actually tell Jenkins what to do:

- `sh`: This executes the shell command; actually, it's possible to define almost any operation using sh.
- `custom`: Jenkins offers a lot of operations that can be used as steps (for example, echo); many of them are simply wrappers over the sh command used for convenience. Plugins can also define their own operations.
- `script`: This executes a block of Groovy-based code that can be used for some non-trivial scenarios where flow control is needed.

The complete specification of the available steps can be found at → here.

Based on the book: „Continuous Delivery with Docker and Jenkins, 3rd Edition - Third Edition By Leszko“
