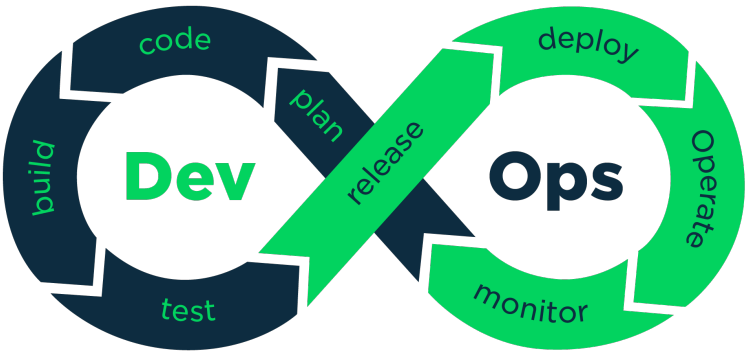
B4 - DevOps

B-DOP-400

my_marvin

Change sides and tame the beast



1.3

my_marvin

- The totality of your source files, except all useless files (binary, temp files, obj files,...), must be included in your delivery.

Jenkins is the most popular Open Source automation platform.

It can automate any task you want, from simple periodic cleanups to full production-scaled deployments. Therefore, it allows to setup the cornerstone of the DevOps chain: the Continuous Integration / Continuous Delivery (or Continuous Deployment) pipeline.
You might also have already read or heard about it, going by the name of *CI/CD pipeline*.

During this project, you are going to master the art of automating your everyday developer tasks with Jenkins.

Did you know that the entire Automated Testing system at Epitech is made with Jenkins? It can efficiently test close to 50 different students at the same time, and in a short period of time (unless your program is stuck in an infinite loop, but that is your fault).

Technical formalities

The project will be entirely evaluated with Automated Tests. Marvin evaluating your Marvin, does not that sound nice? Automatically testing via an automation platform the automatic configuration of an automation platform… thanks DevOps! ;)

In order to be correctly evaluated, you need to carefully read the following formalities.

Configuration as Code

Throughout this project you will use Configuration as Code (a.k.a. JCasC), which allows you to describe your entire desired Jenkins configuration in just a single YAML file.

As you might have guessed now, a lot of the developer side of the DevOps universe is really just configuration files. You can even automate the automation platform!

You will have to turn in a YAML file called `my_marvin.yml`, located at the root of the repository, containing all the necessary configuration described below.

There must be no hardcoded password, all of them must be set by retrieving the values of the associated environment variables.
Any violation will result in a failure of the entire project. You have been warned.

Job DSL

You will also have to use the Job Domain Specific Language (a.k.a. Job DSL) in order to create elements such as jobs.

All your DSL scripts must be centralized into one `job_dsl.groovy` file located at the root of the repository. Other Job DSL sources will not be evaluated.

Jenkins and plug-ins' versions

The Jenkins version used for the tests will be the current LTS version.
The plug-ins will be in their latest compatible version with the current LTS version.

Do not use unnecessary plug-ins, as the virtual Jenkins instance will only have the required ones installed. If you use unnecessary plug-ins, the entire DSL correction will fail. You have been warned.

Installed plug-ins: `cloudbees-folder`, `configuration-as-code`, `credentials`, `github`, `job-dsl`, `script-security`, `structs`, `role-strategy` and `ws-cleanup`.

Now, onto the practical specifications!

Configuration specifications

If particular settings or elements are not specified or addressed in the subject, you are free to do as you please with them.

Global configuration

- The instance must be configured with a system message saying "`Welcome to the Chocolatine-Powered Marvin Jenkins Instance.`"
- The *Agent -> Master Access Control* must be enabled.

Users

- Signing up must be disallowed.
- A user named *Hugo* must be created and has:
  - an id `chocolateen`;
  - a password given by the `USER_CHOCOLATEEN_PASSWORD` environment variable.
- A user named *Garance* must be created and has:
  - an id `vaugie_g`;
  - a password given by the `USER_VAUGIE_G_PASSWORD` environment variable.
- A user named *Jeremy* must be created and has:
  - an id `i_dont_know`;
  - a password given by the `USER_I_DONT_KNOW_PASSWORD` environment variable.
- A user named *Nassim* must be created and has:
  - an id `nasso`;
  - a password given by the `USER_NASSO_PASSWORD` environment variable.

Authorization strategy

- The authorization strategy must be role-based.
- A global role named *admin* must be created and:
  - has a "`Marvin master`" description;
  - has all the permissions;
  - is assigned to Hugo.

Do not blindly list all the permissions, find the (single) right one to give.

- A global role named *ape* must be created and:
  - has a "`Pedagogical team member`" description;
  - can build a job and see their workspaces;
  - is assigned to Jeremy.
- A global role named *gorilla* must be created and:
  - has a "`Group Obsessively Researching Innovation Linked to Learning and Accomplishment`" description;
  - has the same permissions as the *ape* role, plus:
    - the ability to create, configure, delete and move a job,
    - cancel builds;
  - is assigned to Garance.
- A global role named *assist* must be created and:
  - has a "`Assistant`" description;
  - can only view jobs and their workspaces;
  - is assigned to Nassim.

- The specified permissions are the only ones to grant, any non-specified permission must not be given (unless it is inherently needed).

- The specified roles are the only ones to define, no other role than those can be defined.

Folder

Tools

- Is named *Tools*.
- Is at root.
- Has a "`Folder for miscellaneous tools.`" description.

Jobs

Each of the following jobs is expected to be enabled and to be a freestyle job.

CLONE-REPOSITORY

- Is named *clone-repository*.
- Is in the *Tools* folder.
- Has a `GIT_REPOSITORY_URL` string parameter with a "`Git URL of the repository to clone`" description and no default value.
- When executed, clone with Git the repository at the specified URL, using a single shell command.
- Performs a pre-build workspace cleanup.
- Is only executed manually.

SEED

- Is named *SEED*.
- Is in the *Tools* folder.
- Has the following string parameters with no default values:
  - `GITHUB_NAME` with a "`GitHub repository owner/repo_name (e.g.: "EpitechIT31000/chocolatine")`" description;
  - `DISPLAY_NAME` with a "`Display name for the job`" description;
- When executed, create a job with the specifications listed below, using a single `job_dsl.groovy` file script execution.
- Is only executed manually.

JOBS CREATED BY THE SEED JOB

The following specifications apply to all the jobs that are to be created by the SEED job.

- Are at root.
- Are named according to the value of the `DISPLAY_NAME` parameter.
- Have a GitHub project property pointing at the repository specified by the value of the `GITHUB_NAME` parameter.
- Have no parameters.
- Are only executed either by:
  - a SCM poll triggered every minute (which starts a build if there are changes since the last one);
  - manual trigger.
- Use the prebuilt Git SCM system to automatically get the GitHub repository given by the value of the `GITHUB_NAME` parameter.
- Perform a pre-build workspace cleanup.
- When executed, launch each of the following commands in **separate** shell script steps:
  - `make fclean`
  - `make`
  - `make test`
  - `make clean`

You can setup both the GitHub project URL and the Git SCM URL with a single instruction using the `GITHUB_NAME` parameter, simplify your job by trying to find it!

A root element is an element visible on the home page's dashboard.

Tips and tricks

- You MUST **test your configuration and your Job DSL scripts** in order to verify that they are working as intended, you must NOT just put something in your files and wait for the Automated Tests to run. If you are *that* lazy, you will at 100 % fail the project, and you will have 31 years of bad luck.
- To facilitate your tests, it is highly encouraged that you **use the first DevOps technology you discovered**; it is WAY more easier to use boxes instead of just manually installing Jenkins on your computer. Wait, they are not called *boxes*? Argh, this whale made me forget their real name…
- **Use your other Epitech projects** to test your *SEED* job, this is what they are for. ;)