Akademia Górniczo-Hutnicza

Wydział Fizyki i Informatyki Stosowanej

Setting up a Jenkins Pipeline for workflow automation

Daria Kokot i Jakub Baran

Setting up a Jenkins Pipeline for workflow automation

Daria Kokot i Jakub Baran

December 18, 2023

Contents

1 Introduction

In this article, we will briefly describe some tools used in the process of automating software development. We will also provide instructions on how to set up a simple project using these tools.

2 Tools Overview

2.1 DevOps

DevOps is the merger of development and operations. It is an approach that encourage people to collaborate to speed up application delivery. DevOps is about getting people to work togheter, break down boundaries between them.

It may be thought as a continuous loop of the SDLC phases. DevOps consist of 5 phases itself. These are:

- Continuous Development
- Continuous Integration
- Continuous Deployment
- Continuous Testing
- Continuous Monitoring

Continuous Development - impoves software quality and security by providing that every change of the software can be tested and verified.

Continuous Integration - the purpose of this phase is to detect software vulnerabilities early so the change or re write of the code will not be as expensive. It is done by incorporating code updates to the central repository and running automatic test.

Continuous Deployment - ensures that software is automaticly realesed after passing the tests.

Continues Testing - it is the part of integration and deployment. It is a method of testing software on every phase of development process.

Continuous Monitoring - is about monitoring security risks and compliance issues at every step of SDLC.

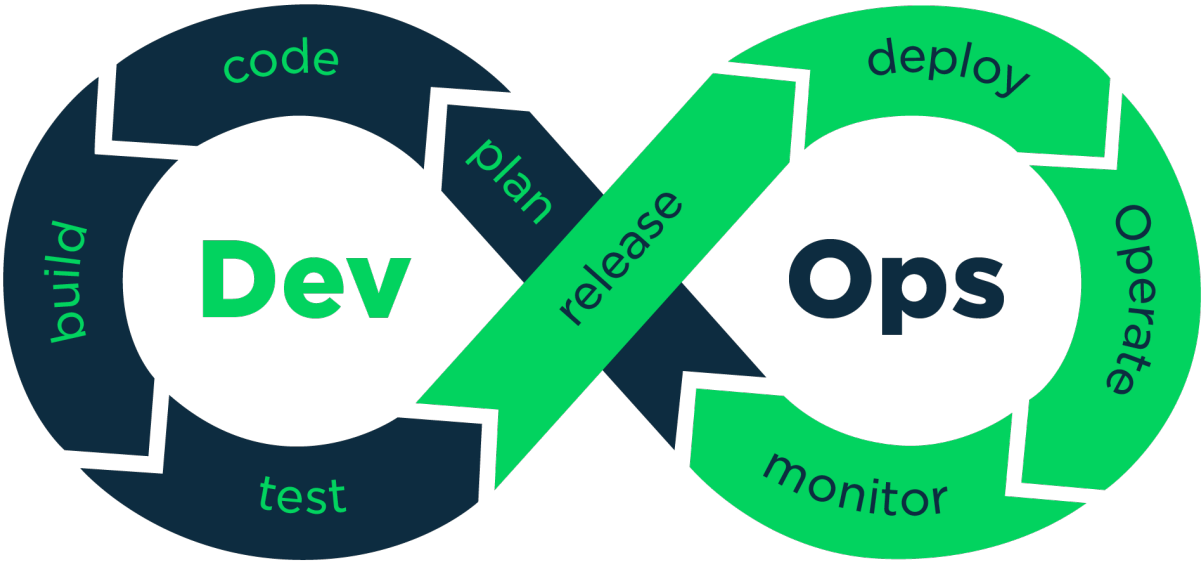

Figure 1: Enter Caption

Popular DevOps tools are:

- Git
- Jenkins
- Ansible
- Docker
- Puppet
- Selenium
- Maven

2.2 Git

What is Git?
Git is a distributed version control system. It allows you to track changes to your source code as you work on a project (usually IT). It is widely used by developers around the world. Platforms that allow the use of the Git system include: GitHub, GitLab, BitBucket. The Git system allows the creation of repositories that store the complete history of a project and all its changes. You can clone a repository, work on it locally, and then send the changes to the main repository. It is also possible to create branches, which provides the ability to work on different functionalities or fixes in parallel. Branches can then be merged to create one consistent version of the project.

How does Git store data and information of changes?
Git treats data like a series of snapshots of a miniature file system. Each time you commit a change or save the state of a project, Git creates a snapshot of all the files. If the files haven't changed, Git just stores references to previous versions, instead of creating new ones. The entire project is, in a way, an ordered list of snapshots.

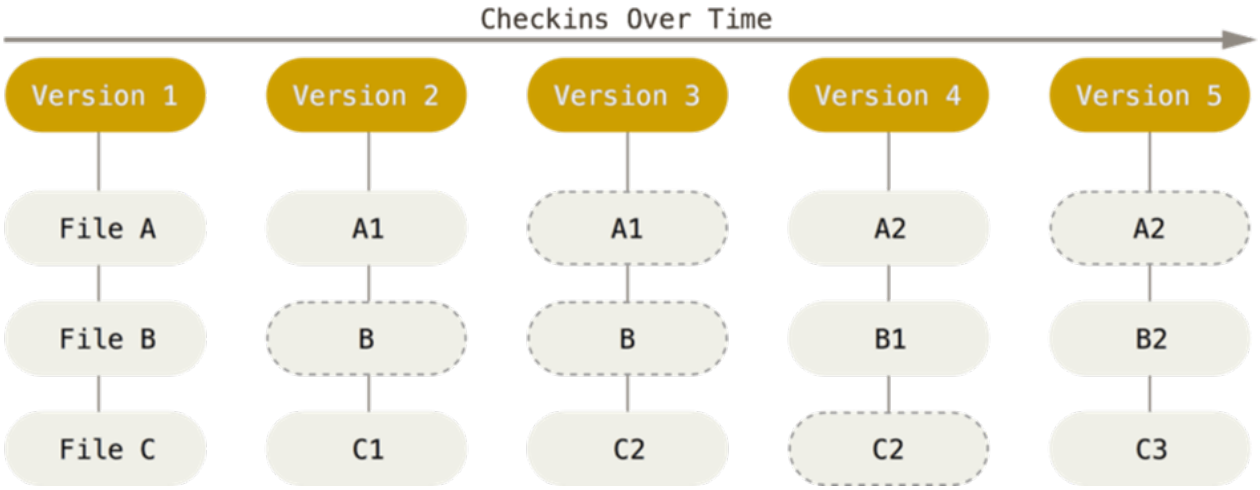

Figure 2: Storing data as snapshots of the project over time

Working locally
Almost all Git operations need only local files and resources. The entire project history is on the local disk, which greatly speeds up all operations. Data is loaded directly from the local database. You don't need an Internet connection or VPN to work with the git system, as long as you don't want to send data to an Internet platform.

Integrity
It is possible to distinguish three states in which the file is located:

- modified - the file is changed, but the change has not been approved
- staged - the file has been staged and will be approved in the next commit
- the file has been safely saved in the local database

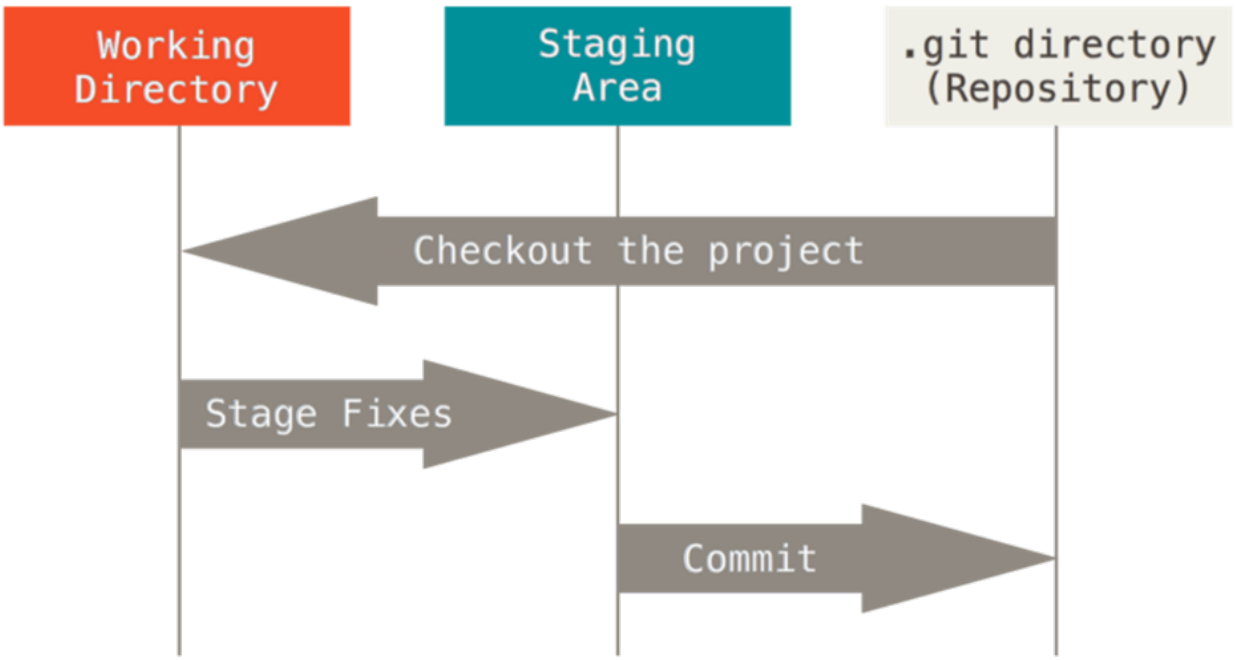

Figure 3: Working directory, staging area, .git directory

The working directory is the folder where you locally work on a project. The staging area is a file containing information about what goes into the next commit. The .git directory is where the project's metadata and object base are stored. The .git folder is created when cloning a project, among other things.

2.3 Jenkins

What is Jenkins?
Jenkins is an open source server for automation. It provides hundreds of plugins to support building, deploying and automating any project.

Applications
Jenkins server can be used for projects written in various programming languages. This server allows you to automate procedures related to building, testing and documenting code, which makes it easier to release any changes in environments (Development, Stage, Production).

CI/CD
Continuous Integration (CI), Continuous Delivery (CD) and Continuous Deployment (CD) are key concepts used in Jenkins Server. Continuous integration refers to the frequent uploading of code snippets to the repository. The code is compiled and tests are run. In case of an error, feedback is returned. This mechanism reduces the number of errors and the cost of finding them. Continuous delivery is possible automatically after the integration and testing phase. It involves preparing the application, packaging it and placing it in a central repository. Continuous deployment allows the application to be automatically installed on the production server.

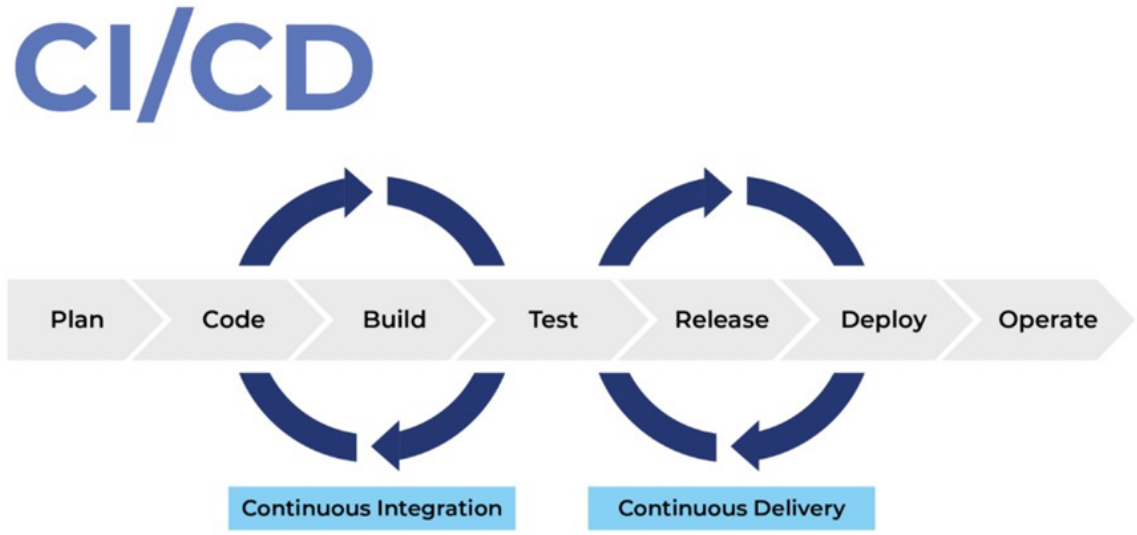

Figure 4: Continuous Integration (CI), Continuous Delivery (CD) and Continuous Deployment (CD)

Jenkins architecture

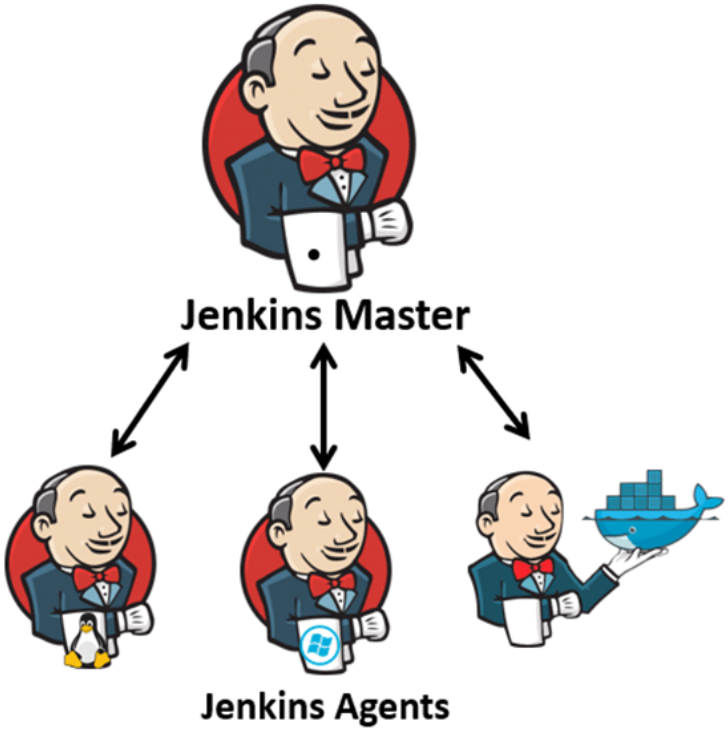

Figure 5: Jenkins architecture

The Jenkins server uses a master/slave architecture. This means that the central server manages the operations of slave machines. Let's call the slave instances agents. They can be on different machines (servers, virtual machines, microcomputers, etc.) having different operating systems. Agents are responsible for performing various tasks such as compiling, running tests and creating application packages (application packaging). Let's call the master instance a controller. This is the server that regularly monitors the status of each connected agent. It checks the network connection, disk occupancy, RAM availability and other relevant statistics. It controls the delegation of tasks to the appropriate agents and makes sure that no slave machine is over-tasked. The controller handles user interface, configuration storage and user account management. It can also archive files while running tasks as test reports. This architecture provides flexibility. You can easily increase hardware resources as the task queue increases.

Jobs

Using Jenkins, you can define jobs that describe how to compile, test and deploy an application. These jobs are run automatically at three stages of application preparation: building, testing, deploying. A configured sequence of events consisting of jobs is always executed in the same way. Jobs can be prepared on both using the GUI and a configuration file. Jobs using the configuration file are called "pipeline". Defined jobs can be run. A single run is called "build". During this process, activities such as building, testing, etc. are carried out sequentially. Build can end with one of the following statuses:

- success - all steps and tests performed correctly
- unstable - all steps performed correctly, but some tests return errors

- failure - an error occurred during the execution of the task (e.g. unavailable server build or incorrect configuration parameters)

The results of running a build are stored on the master server (controller). This makes it possible to track the history of builds. The task execution process can be observed from the user interface. Tasks can be set to be performed periodically.

Plugins
Jenkins server can be extended by adding plugins. This allows integration with external tools such as:

- Jira for project management
- GitHub, BitBucket, GitLab for version control integration
- AWS EC2 for managing Amazon Web Services (AWS) cloud instances directly from Jenkins
- plug-ins to extend the possible steps to execute in a task (job)
- and many, many more

2.4 AWS EC2

What is Amazon EC2?
Amazon Elastic Compute Cloud (Amazon EC2) is an IaaS (Infrastructure as a Service) cloud computing service. It allows users to rent virtual machines (called instances) in the cloud. These services can be configured and scaled as needed. Amazon EC2 provides scalable computing power in the Amazon Web Services (AWS) cloud. Using this service greatly eliminates the need for upfront investment in hardware, so you can develop and deploy applications faster. It allows you to run as many or as few virtual servers as you want, configure security and networking, and manage storage. Amazon EC2 allows you to scale up or down to handle changes in requirements or spikes in popularity, reducing the need for traffic forecasting. Amazon EC2 architecture The diagram below shows the basic architecture of an Amazon EC2 instance deployed as part of Amazon's virtual private cloud (VPC) located in the region's availability zone. A virtual firewall, which is a security group that controls incoming and outgoing traffic. The private key is stored on the local computer, and the public key is stored on the instance. These keys are defined by a pair of keys confirming user identities. In this example, the instance is served by an Amazon EBS volume. The VPC communicates with the Internet using an Internet gateway.

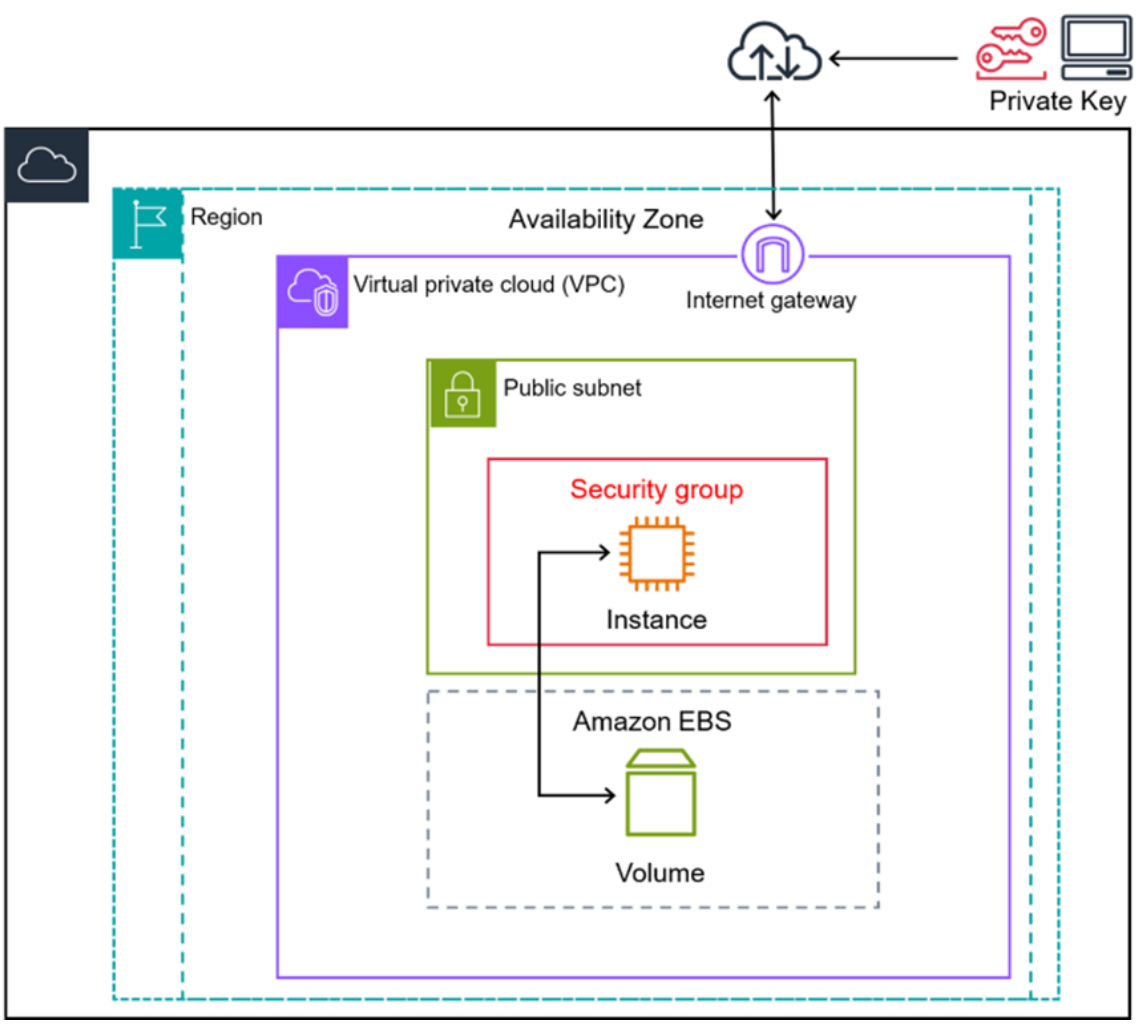

Figure 6: Basic architecture of an Amazon EC2 instance deployed as part of Amazon's virtual private cloud (VPC)

Amazon EC2 features

Amazon EC2 provides the following functions: - Creating instances, which are flexible virtual computing environments that can be configured and run as needed - Creation of Amazon Machine Images (AMIs), which are pre-built templates that contain the operating system and additional software to facilitate rapid instance creation - The ability to choose different configurations of CPU, memory, storage and network bandwidth for instances, allowing you to tailor resources to specific tasks - Verification when accessing instances using key pairs, which provides a secure way to log in; AWS stores the public key and the private key is stored locally by the user - Temporary and persistent storage - Multiple physical resource locations (Amazon EBS instances and volumes-so-called Regions and Availability Zones) - Configurable security groups that provide the ability to specify protocols, ports and IP address sources that can access instances - Flexible IP addresses that can be dynamically assigned to resources in the cloud, thus making them easier to manage - Ability to create and assign tags to resources - Virtual private clouds (VPCs)

Instance types

Based on the requirements of the application or software, the type of the instance is determined during startup. This defines the hardware of the host computer used by the instance. This defines distinct hardware capabilities and a consistent and predictable amount of CPU processing power, regardless of the hardware you are physically running on. Amazon EC2 allocates certain host computer resources such as CPU, memory and instance storage to a specific instance. Host computer resources, such as network and disk subsystem, are shared between instances. Each type of instance in Amazon EC2 offers different levels of I/O (input/output) performance, which determines both the minimum and maximum performance of available resources. This allows you to adjust performance depending on your application requirements. This provides greater control over system performance in terms of data input/output operations.

3 Instruction

In this section, we'll offer guidance on setting up a simple Jenkins pipeline and configuring it to automatically initiate the pipeline whenever someone commits to a Git repository. We will create a simple web page to demonstrate the entire process.

- **Step 1 - Setting up AWS EC2 Jenkins server**:First, let's begin by creating a new AWS EC2 instance, preferably using the Red Hat distribution. In the instance's network settings, open port 8080 for internet access and port 22 for SSH connections.

  Once the instance is running, establish an SSH connection and install Java by following the instructions provided on the OpenJDK installation page.

  Next, proceed to install Jenkins using the following commands:

  ```
  sudo yum -y install wget
  sudo wget -O /etc/yum.repos.d/jenkins.repo https://pkg.jenkins.io/
      redhat-stable/jenkins.repo
  sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io.
      key
  sudo yum -y install jenkins
  sudo systemctl start jenkins
  ```

  Now that your Jenkins server is up and running, you can access it via a web browser. Remember, use 'http' instead of 'https.' Simply enter your server's public IP address followed by port 8080, like this:

  `http://51.20.130.121:8080`

- **Step 2 - Creating a Github repository**: This step is straightforward. Create an account on the webpage https://github.com/ and establish your first repository. Initially, add a simple webpage to it. Feel free to use our repository if you prefer: https://github.com/z3r0tru5t/DevOps.

- **Step 3 - Setting up AWS EC2 web page server**: In this step, you need to create another EC2 instance and open ports 22 and 80 to access your website via a browser. Connect to the server using SSH and execute the following commands:

  ```
  sudo su
  yum install git
  yum install httpd -y
  amazon-linux-extras install php8.0 -y
  service httpd start
  service httpd status
  cd /var/www/html
  git init
  git pull https://github.com/YOUR_USERNAME/YOUR_REPOSITORY.git
  ```

  After executing these commands, when you enter the following URL in your browser:

  `http://YOUR_IP:80`

  you should be able to view your webpage.

- **Step 4 - Configuring Jenkins pipeline**: Now that your environment is set up, let's create a pipeline. Open your Jenkins server web application. Create a new project and choose 'Pipeline'. Enable 'GitHub hook trigger for GITScm polling' so that our project triggers the pipeline after every commit.

  In the pipeline configuration, you can choose whether to have your pipeline file in the repository or in Jenkins. We've added ours to the repository, enabling automatic deployment with commits. The pipeline should involve steps like cloning your repository and pushing changes to the web server. You'll find these steps outlined in the Jenkinsfile in our repository. Feel free to add additional steps like running various tests, etc.

  Once your pipeline is set up, create a webhook in the GitHub repository settings. Set the Payload URL to: `http://YOUR_JENKINS_SERVER_IP:8080/github-webhook/`.

  Now, go to your profile settings in GitHub and, in the developer settings, generate a new personal access token. In Jenkins, navigate to 'Manage Jenkins', 'System' and add a GitHub server. Choose 'Add Credentials', then in 'Kind' select 'Secret Text' and paste your access token into the 'Secret' field. You can write whatever you prefer in the description and ID. Test the connection; if it's not working, ensure you copied your access token correctly.

  When the connection is successful, after every commit in your main branch, Jenkins should automatically start the pipeline.

Creating your first Jenkins pipeline and automating project deployment involves several steps. The initial setup process can be time-consuming, but once configured, it becomes a versatile tool usable across multiple projects.

References

[1] https://git-scm.com/book/en/v2/Getting-Started-What-is-Git%3F

[2] https://szkolajenkinsa.pl/2021/02/07/czym-jest-jenkins/

[3] https://blurify.pl/blog/czujny-pan-jenkins-czyli-troche-o-ci-w-studio-software/#Budowanie_CD_z_Jenkins

[4] https://blog.devops.dev/best-practices-for-building-a-ci-cd-pipeline-with-jenkins-c

[5] https://www.hostersi.pl/uslugi-chmurowe/czym-jest-amazon-ec2/

[6] https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html