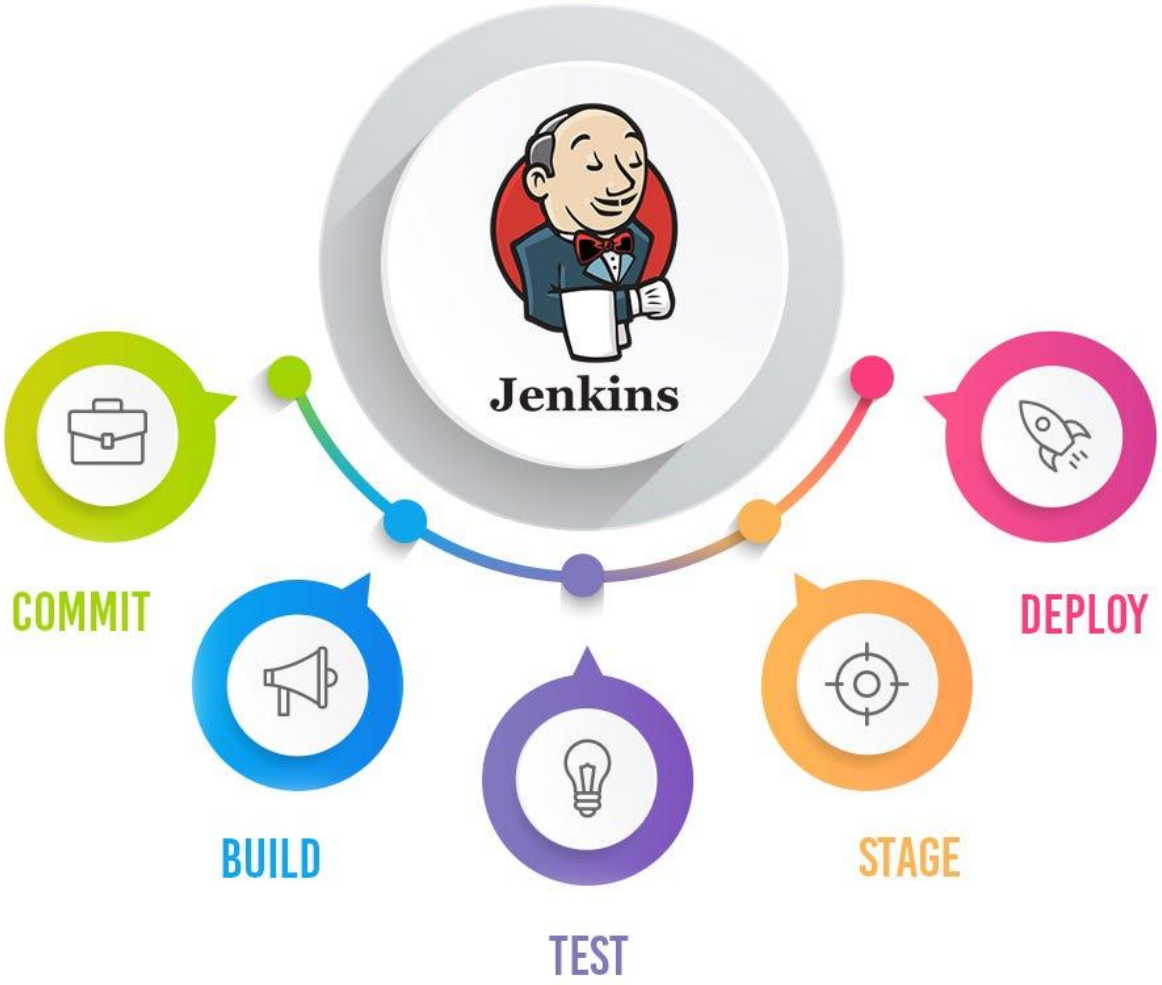
JENKINS



Jenkins is an open-source automation server that facilitates the building, testing, and deployment of software by automating various stages of the software development lifecycle. It is one of the most widely used tools for implementing continuous integration (CI) and continuous delivery (CD) practices. Here's an explanation of key concepts and features of Jenkins:

1. **Continuous Integration (CI):**
    - Jenkins promotes the practice of continuous integration, where developers regularly integrate their code changes into a shared repository. This helps detect and address integration issues early in the development process.
2. **Automation Server:**
    - Jenkins serves as an automation server that can be extended and customized through plugins. It automates repetitive tasks, such as building code, running tests, and deploying applications, allowing development teams to focus on writing code.
3. **Jenkins Master and Agent:**
    - Jenkins has a master-slave architecture. The Jenkins master is the primary server responsible for managing jobs, scheduling builds, and monitoring agents. Agents (or slaves) are distributed nodes that execute tasks as instructed by the master.
4. **Jobs and Pipelines:**
    - In Jenkins, a job is a single task or unit of work. Jobs can be configured to perform various activities, such as compiling code, running tests, and deploying applications. Jenkins pipelines provide a way to define complex workflows as code, allowing the creation of end-to-end delivery pipelines.
5. **Plugins:**
    - Jenkins supports a vast ecosystem of plugins that extend its functionality. Plugins are used to integrate Jenkins with version control systems (e.g., Git), build tools, deployment platforms, and various other tools. They enable users to tailor Jenkins to their specific needs.

6. **Jenkinsfile:**
    - A Jenkinsfile is a text file that defines a pipeline as code. It is typically stored in the version control repository along with the application code. Jenkinsfiles provide a way to define and manage pipeline configurations, making it easier to version control and reproduce builds.
7. **Build and Deployment:**
    - Jenkins can automate the build process, compiling source code into executable artifacts. It can also facilitate the deployment of applications to various environments, ensuring a consistent and repeatable process.
8. **Monitoring and Notifications:**
    - Jenkins provides monitoring capabilities, allowing users to track the status of builds and pipelines. Notifications can be configured to alert teams about build failures or other issues, promoting quick resolution.
9. **Extensibility:**
    - Jenkins is highly extensible, allowing users to customize it to fit their specific requirements. The extensive plugin ecosystem and support for scripting languages like Groovy contribute to its flexibility.
10. **Community Support:**
    - Jenkins has a vibrant and active community that contributes to its development and provides support through forums, documentation, and a wealth of shared knowledge.

Jenkins plays a crucial role in modern software development by automating repetitive tasks, improving collaboration among development teams, and facilitating the delivery of high-quality software.

Install Jenkins in ec2 instances:

Prerequisites:

1. **EC2 Instance:**
    - Launch an EC2 instance using the AWS Management Console.
    - Ensure that the security group associated with your EC2 instance allows inbound traffic on port 8080 (or the desired port for Jenkins).
2. **SSH Access:**
    - Connect to your EC2 instance using SSH.

Steps:

Downloading and installing Jenkins

Completing the previous steps enables you to download and install Jenkins on AWS. To download and install Jenkins:

1. Ensure that your software packages are up to date on your instance by using the following command to perform a quick software update:

   [ec2-user ~]$ sudo yum update –y

2. Add the Jenkins repo using the following command:

   [ec2-user ~]$ sudo wget -O /etc/yum.repos.d/jenkins.repo \

   https://pkg.jenkins.io/redhat-stable/jenkins.repo

3. Import a key file from Jenkins-CI to enable installation from the package:

   [ec2-user ~]$ sudo rpm --import https://pkg.jenkins.io/redhat-stable/jenkins.io-2023.key

   [ec2-user ~]$ sudo yum upgrade

4. Install Java (Amazon Linux 2023):

   [ec2-user ~]$ sudo dnf install java-17-amazon-corretto -y

5. Install Jenkins:

   [ec2-user ~]$ sudo yum install jenkins -y

6. Enable the Jenkins service to start at boot:

[ec2-user ~]$ sudo systemctl enable jenkins

7. Start Jenkins as a service:

   [ec2-user ~]$ sudo systemctl start jenkins

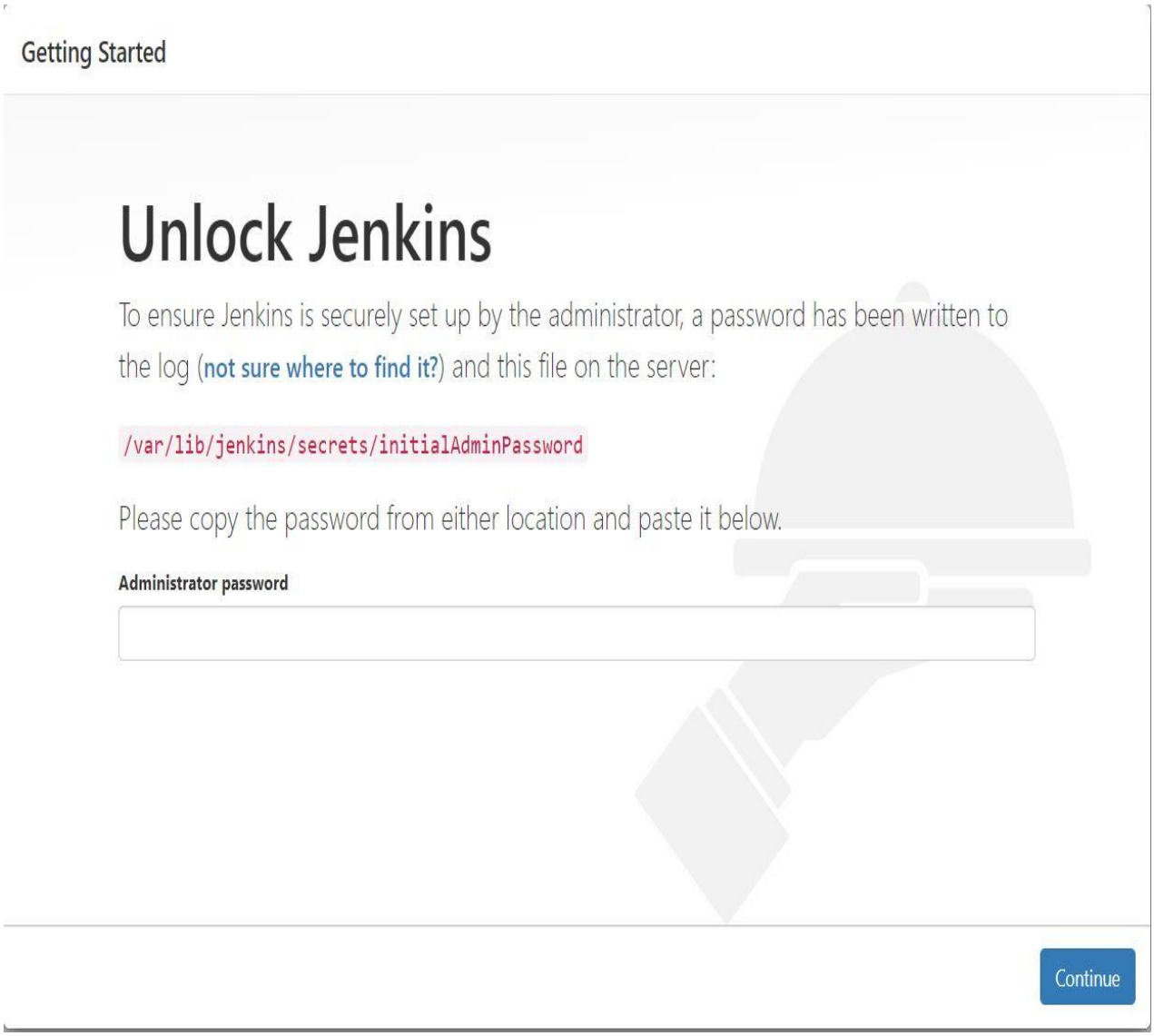
8. You can check the status of the Jenkins service using the command:

   [ec2-user ~]$ sudo systemctl status jenkins

Configuring Jenkins

Jenkins is now installed and running on your EC2 instance. To configure Jenkins:

1. Connect to http://<your_server_public_DNS>:8080 from your browser. You will be able to access Jenkins through its management interface:



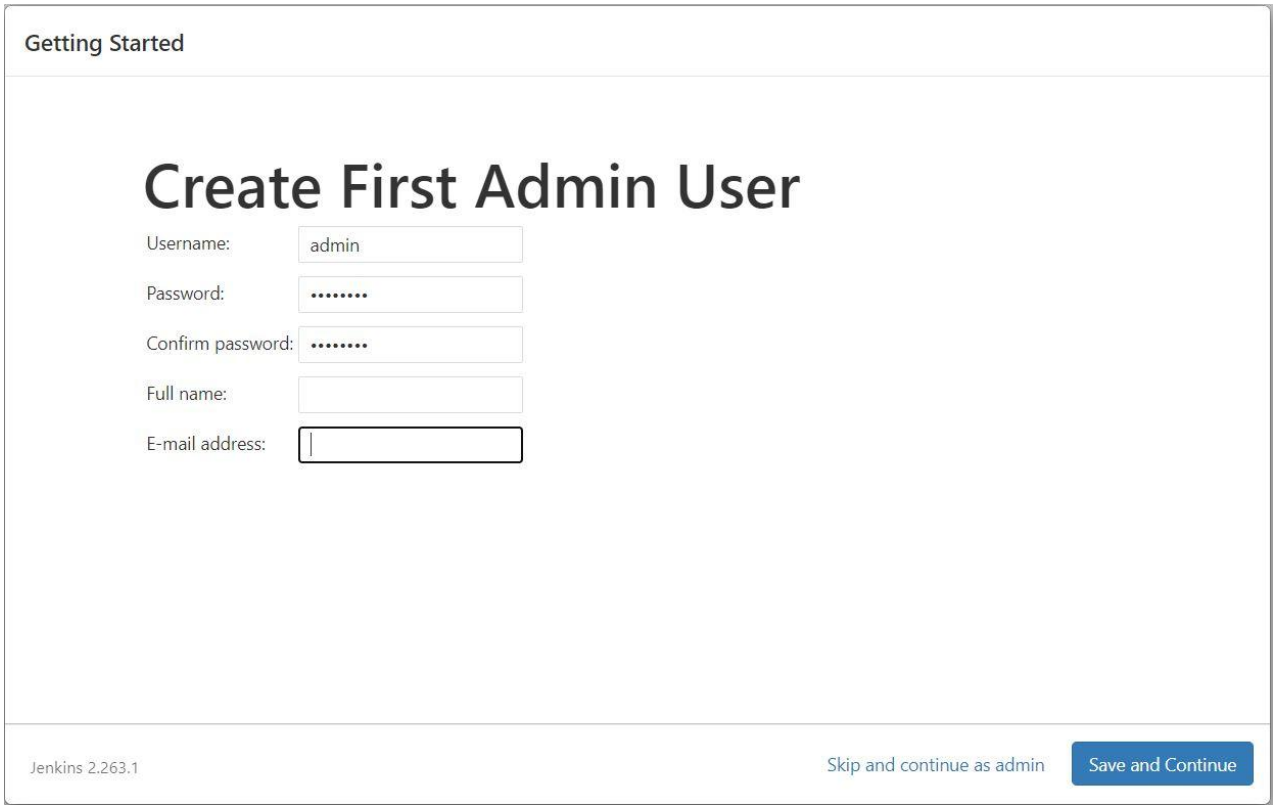
2. As prompted, enter the password found in **/var/lib/jenkins/secrets/initialAdminPassword**.

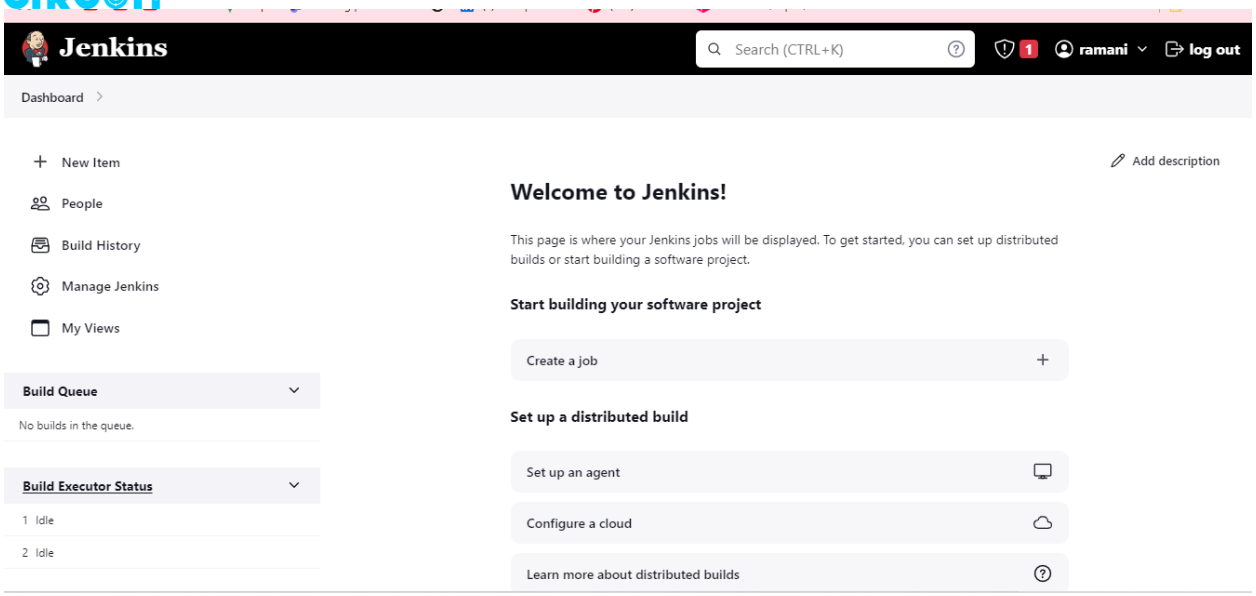
    a. Use the following command to display this password:

    [ec2-user ~]$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword

3. The Jenkins installation script directs you to the **Customize Jenkins page**. Click **Install suggested plugins**.

4. Once the installation is complete, the **Create First Admin User** will open. Enter your information, and then select **Save and Continue**.





Steps to Create an Empty Job (Freestyle Project):

1. **Login to Jenkins:**
    - Open your web browser and navigate to Jenkins. Usually, it's at **http://your_jenkins_server:8080**.
    - Log in with your credentials.
2. **Access Jenkins Dashboard:**
    - After logging in, you'll be on the Jenkins dashboard.
3. **Create a New Job:**
    - Click on "New Item" or "Create a job" on the Jenkins dashboard.
4. **Enter Job Name:**
    - Enter a name for your job in the "Enter an item name" field.
5. **Select Freestyle Project:**
    - Choose "Freestyle project" as the job type.
6. **Configure the Job:**
    - You will be redirected to the configuration page for the new job.
    - The configuration page allows you to define various settings for your job.

- The configuration page includes sections such as "General," "Source Code Management," "Build Triggers," "Build," and more. For an empty job, you can skip most of the sections.

7. **Save the Job:**
    - Scroll down to the bottom of the page and click on the "Save" button.
8. **Build the Empty Job:**
    - Now that you have created the job, you can click on "Build Now" to execute the job manually.
9. **View Job Results:**
    - After the build is completed, you can view the build results and console output by clicking on the build number under the "Build History" section.

Creating a user in Jenkins

Access Jenkins Web Interface:

- Open a web browser and navigate to your Jenkins instance. The default address is usually **http://localhost:8080** unless you've configured a different port or domain.

2. **Log in:**
    - If you haven't logged in yet, you'll be prompted to enter the initial administrative password. You can find this password in the Jenkins server logs or in the file system.
3. **Login as Admin:**
    - Log in with the admin credentials. The default admin username is usually **admin**.
4. **Navigate to "Manage Jenkins" > "Manage Users":**
    - Click on "Manage Jenkins" in the left-hand sidebar, and then click on "Manage Users."
5. **Create a New User:**
    - Click on the "Create User" link.

6. **Fill in User Details:**

- Enter the required user details, including username, password, full name, and email address.

7. **Assign the User to a Group (Optional):**
    - You can assign the user to a group if your Jenkins setup involves user groups. This is optional.
8. **Save the User:**
    - Click the "Create User" button to save the new user.

"Manage Jenkins" is a section within the Jenkins web interface that provides administrative capabilities for configuring and maintaining the Jenkins environment. This section is accessible to users with administrative privileges, typically the initial administrator created during the Jenkins setup.

1. **Configure System:**
    - This option allows administrators to configure global settings for Jenkins. It includes system-related configurations such as security settings, global tool configurations, and other settings that affect the entire Jenkins instance.
2. **Configure Global Security:**
    - Administrators can manage security settings for Jenkins, including authentication methods, authorization strategies, and other security-related configurations.
3. **Manage Plugins:**
    - Jenkins functionality can be extended through plugins. In this section, administrators can install, uninstall, and manage plugins to enhance or customize Jenkins features.
4. **Manage Nodes and Clouds:**
    - Jenkins uses nodes to distribute build and deployment tasks across different machines. This section allows administrators to manage the configuration of these nodes, including adding, configuring, or removing them.
5. **Reload Configuration from Disk:**
    - This option allows administrators to reload the Jenkins configuration from the disk without restarting the entire Jenkins instance. It is useful after manually modifying Jenkins configurations.

6. **System Information:**
    - This section provides information about the Jenkins environment, including the Java Virtual Machine (JVM) system properties, environment variables, and other system-related details.
7. **Script Console:**
    - The Script Console allows administrators to run Groovy scripts directly on the Jenkins server. This can be useful for performing administrative tasks and troubleshooting.
8. **Support:**
    - This section provides information and links for getting support for Jenkins, including accessing documentation, community forums, and reporting issues.
9. **Load Statistics:**
    - Jenkins keeps track of load statistics, including the number of builds executed and the time spent on builds. This information can be useful for monitoring and optimizing Jenkins performance.

Overall, "Manage Jenkins" serves as the central hub for administrators to configure, customize, and manage the Jenkins environment to meet the specific needs of their software development and continuous integration/continuous delivery (CI/CD) processes. It's a critical area for maintaining the health and functionality of a Jenkins instance.

Authentication

Authentication in Jenkins refers to the process of verifying the identity of users or systems trying to access the Jenkins environment. It ensures that only authorized entities can interact with Jenkins, protecting the integrity and security of the continuous integration and continuous delivery (CI/CD) processes. Jenkins supports various authentication mechanisms:

Jenkins Own User Database:

- Jenkins has its own user database where user credentials (username and password) are stored locally.
- Users are created and managed within Jenkins, and authentication is performed against this internal database.

Authorization

Authorization in Jenkins refers to the process of controlling access and permissions within the Jenkins environment. It ensures that users and processes have appropriate levels of access to perform specific actions, protecting the integrity and security of Jenkins configurations and resources. Jenkins provides several features and mechanisms for authorization:

1. **Security Realms:**
    - A security realm in Jenkins determines where Jenkins looks for user information and how it authenticates users.
    - Common security realms include Jenkins' own user database, LDAP (Lightweight Directory Access Protocol), Active Directory, and other external authentication providers.
    - The security realm handles the authentication part of user access.
2. **Matrix-Based Security:**
    - Matrix-based security allows administrators to define permissions using a matrix, where the rows represent users/groups, and the columns represent permissions.
    - Permissions can include configuring jobs, building jobs, deleting jobs, and more.
    - Users or groups can be granted specific permissions for individual tasks or across all tasks.
3. **Project-Based Matrix Authorization:**
    - This is an extension of matrix-based security but applied at the project level.
    - For each project, administrators can define a matrix to grant permissions specifically for that project.
    - Project-based matrix authorization allows for more fine-grained control over who can perform specific actions within a project.
4. **Roles:**
    - Roles provide a way to group users or groups together and assign them specific permissions.
    - Roles simplify the management of permissions by allowing administrators to grant or revoke permissions for an entire group of users.

- Roles can be used in conjunction with matrix-based security or project-based matrix authorization.

Adding users in metric based security authorization

1. **Choose Security Realm:**
    - Under "Security Realm," choose the appropriate authentication method (e.g., Jenkins' own user database, LDAP, etc.).
    - Configure the selected security realm settings.
2. **Choose Authorization:**
    - Under "Authorization," select "Matrix-based security."
3. **Configure Permissions:**
    - In the matrix, you'll see a grid with usernames/groups (rows) and permissions (columns).
    - Check the checkboxes for the desired permissions for each user or group. Common permissions include:
        - Overall - Read (Read access to Jenkins)
        - Job - Build (Ability to build jobs)
        - Job - Configure (Ability to configure jobs)
        - Job - Read (Read access to jobs)
4. **Save Changes:**
    - Scroll down to the bottom of the page and click the "Save" button to apply the changes.

Here's an example matrix:

User/Group	Overall - Read	Job - Build	Job - Configure	Job - Read
admin	☑ ☐	☑ ☐	☑ ☐	☑ ☐
developer	☑ ☐	☑ ☐		☑ ☐
anonymous	☐	☐		☐

In this example:

- The "admin" user has full permissions.
- The "developer" user has read access to Jenkins, the ability to build jobs, and read access to jobs.
- The "anonymous" user has no permissions.

Remember to tailor the permissions based on your specific requirements. The matrix provides a fine-grained control mechanism for assigning permissions to users or groups. After saving the changes, users will have the specified permissions according to the matrix configuration.