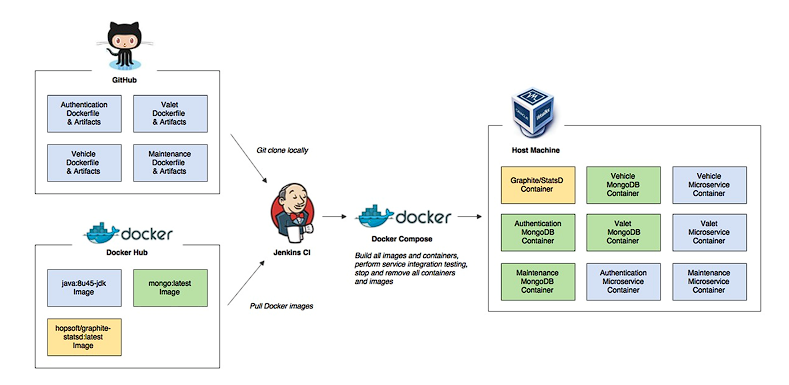
**Batch 2: Group 1**

**Assignment 11**

**Aim:** Create your initial product backlog: Create initial product backlog items as user stories, order your product backlog.

**Objectives:**

1. Implement robust authentication and authorization mechanisms to ensure secure access control within the deployed application.
2. Utilize Jenkins, an automation server, to streamline the deployment process of containerized applications.
3. Enhance deployment reliability and efficiency by integrating authentication and authorization configurations seamlessly into the deployment pipeline.



**1.Authentication:**

* Authentication in Jenkins refers to the process of verifying the identity of users accessing the Jenkins server.
* Jenkins supports various authentication mechanisms, including:
* Jenkins' own user database
* LDAP (Lightweight Directory Access Protocol)
* Active Directory
* OAuth
* To configure authentication, navigate to "Manage Jenkins" > "Configure Global Security" and select the desired authentication method.

Commands:

A screenshot of a computer

Description automatically generated

**2.Authorization:**

* Authorization in Jenkins defines what actions users or groups of users are allowed to perform within the Jenkins environment.
* Jenkins supports several authorization strategies, such as:
* Matrix-based security: Allows administrators to define permissions for individual users or groups on specific Jenkins features.
* Role-based access control (RBAC): Assigns users to roles, each with a predefined set of permissions.
* To configure authorization, navigate to "Manage Jenkins" > "Configure Global Security" and select the desired authorization strategy.

**Deploying an Application to a Container Using Jenkins:**

1. Set up Docker and Docker Plugin:

* Ensure Docker is installed on the Jenkins server and accessible to Jenkins.
* Install the Docker plugin in Jenkins to enable interaction with Docker from Jenkins pipelines.

1. Create a Jenkins Pipeline:

* Define a Jenkins pipeline using a Jenkinsfile that specifies the stages and steps required to deploy the application to a container.
* Include stages for building the application, creating a Docker image, and deploying the image to a container.

1. Configure Docker Credentials:

* If the Docker registry requires authentication, configure Docker credentials in Jenkins.
* Navigate to "Manage Jenkins" > "Manage Credentials" to add Docker registry credentials.

1. Write Deployment Script:

* Write a deployment script within the Jenkins pipeline to interact with Docker.
* Use Docker commands or Docker API to build the Docker image and push it to the Docker registry.

1. Execute Pipeline:

* Run the Jenkins pipeline to execute the deployment process.
* Jenkins will execute the defined stages and steps, including building the application, creating the Docker image, and deploying it to a container.

1. Verify Deployment:

* Once the pipeline execution is complete, verify that the application has been successfully deployed to the container.
* Access the deployed application to ensure it is running as expected.

By following these steps, users can effectively manage authentication and authorization in Jenkins and deploy applications to containers using Jenkins pipelines, thereby streamlining the deployment process and enhancing productivity in software development workflows.

URL: <https://programmaticponderings.com/2015/06/22/continuous-integration-and-delivery-of-microservices-using-jenkins-ci-maven-and-docker-compose/>

**Conclusion:**

Integrating robust authentication and authorization mechanisms into the deployment pipeline using Jenkins ensures that deployed applications are securely accessed and protected. By automating the deployment process with Jenkins, organizations can achieve faster and more reliable deployments of containerized applications. This approach not only enhances security but also improves deployment efficiency, allowing teams to focus more on delivering value to end-users. Overall, combining authentication, authorization, and automated deployment using Jenkins contributes to a more secure, efficient, and scalable software delivery process.