Docker-based Jenkins quickstart examples

Google Summer of Code Program 2023 Project Proposal

Arnav Gupta
Arnav.gupta.2003@gmail.com
Github.com/arnavgupta2003

Project Abstract.

The documentation for using Docker with Jenkins is currently too difficult for beginners. The author proposes creating a set of docker-compose files for various types of Jenkins instances, described in the documentation and tested weekly through ci.jenkins.io. The documentation would use the Include Content by URI feature to ensure that code snippets are up-to-date and tested by ci.jenkins.io. Additionally, users could run these code snippets using GitPod.

Project Description.

The Docker-based Jenkins quickstart examples project aims to provide a simple and easy way for beginners to start a local Jenkins instance using Docker. The project includes a set of docker-compose files representing various types of Jenkins instances, such as simple Docker and Kubernetes.

The project's main focus is on providing users with step-by-step instructions on how to use the docker-compose files to start a Jenkins instance. The documentation will include code snippets tested weekly using ci.jenkins.io to ensure they are up-to-date and working correctly.

The main motivation for developing quick start examples is to ensure that the benefits of Jenkins are extended to the common consumer. By providing the updated documentation, the users can set up Jenkins environments for various purposes quickly.

With help from the Jenkins community, we can provide robust docker-compose files, a perfect solution for both basic and advanced users.

By contributing to this project, I can help beginners and other users looking for a simple and easy way to start a local Jenkins instance using Docker. My contributions will make it easier for them to start with Jenkins and contribute to the community.

Steps:

Developing a project to provide a simple way to start with Jenkins and Docker involves several steps. Here's a detailed guide on how this project will follow:

- **Identify the Requirements**: The first step is identifying the project's requirements. This involves determining the types of Jenkins instances that need support and the Docker images required for each instance. It also involves identifying the tools and technologies required to develop, test, and deploy the project.

- **Design the Architecture**: The next step is to design the project's architecture. This involves designing the system's components, such as the Docker images, Jenkins instances, and the pipeline for continuous integration and deployment. It also involves designing the architecture to ensure scalability, performance, and security.

- **Develop the Docker-Compose Files**: After designing the architecture, the team must develop the Docker-compose files representing various types of Jenkins instances. Each Docker-compose file should contain the necessary components, such as the Jenkins image, plugins, and configurations. It should also include a description of the Jenkins instance, including its purpose, features, and requirements.

- **Develop the CI/CD Pipeline**: The next step is to develop the continuous integration and deployment pipeline. The pipeline should be designed to automatically build, test, and deploy the Docker images to production. It should also include testing the Docker-compose files weekly using ci.jenkins.io to ensure they are still working.

- **Develop the Documentation**: Once the Docker-compose files and CI/CD pipeline are developed, the next step is to develop the documentation. The documentation should describe the project's purpose, how to get started, and how to use the various Jenkins instances. The documentation should also describe the Docker-compose files and how to use them.

- **Test and Deploy the Project**: The final step is to test and deploy the project. The team should conduct unit, integration, and system testing to ensure the project works as expected. Once testing is complete, the team can deploy the project to production.

- **Maintain and Update the Project**: After deploying the project, the team needs to maintain and update it regularly. This involves ensuring that the documentation is up-to-date, fixing bugs and issues, and updating the Docker images, plugins, and configurations to ensure compatibility and security.

Preview of the Steps:

- **Identify the Requirements**:
    1. Compatibility: The project should support different versions of Docker and Jenkins, and other related technologies.
    2. Simplification: The project should simplify setting up Jenkins with Docker for beginners, making it easy to use.
    3. Customizability: The project should provide a way to customize Jenkins instances according to specific requirements, such as plugins, configurations, and themes.
    4. Testing: The project should include testing and validation of the Docker-compose files and Jenkins instances to ensure that they work correctly. (Using ci.jenkins.io)
    5. Documentation: The project should include clear and comprehensive documentation that explains the purpose, features, and usage of each Jenkins instance, along with instructions on how to get started.
    6. Security: The project should ensure that the Docker containers are secure, and best practices for securing Docker containers should be followed.
    7. Scalability: The project should be scalable, supporting an increasing number of users and Jenkins instances.
    8. Automation: The project should automate the building, testing, and deployment of Docker images to production. (Github Actions)
    9. Support: The project should support users who encounter issues or have questions while using Jenkins with Docker. (Jenkins Community)

    Overall, the requirements for this project focus on providing a simple, customizable, and secure way to get started with Jenkins and Docker while ensuring scalability, automation, and support.

- **Develop the Docker-Compose Files**
  docker-compose.yml

```
version: '3.3'
services:
    jenkins:
        ports:
            - '2376:80'
        image: jenkins/jenkins
```

- **Develop the CI/CD Pipeline**
  Jenkins File:

```
pipeline {
    agent {
        docker {
            image 'jenkins/jenkins'
            args '-p 2376:80'
        }
    }
    stages {
        stage('Build') {
            steps {
                sh 'docker run -p 2376:80 jenkins/jenkins'
            }
        }
        stage('Test') {
```

```
            steps {
                //Testing steps
            }
        }
        stage('Deploy') {
            steps {
                //Deploying Steps
            }
        }
    }
}
```

Github Actions YML:

```
name: Jenkins Pipeline
on: [push]
jobs:
  build:
    runs-on: ubuntu-latest
    steps:
    - name: Checkout code
      uses: actions/checkout@v2
    - name: Start Jenkins container
      run: docker run -d -p 2376:8080 jenkins/jenkins
    - name: Wait for Jenkins to start
      run: sleep 30
    - name: Build stage
      run: docker exec -it $(docker ps -q --filter
"ancestor=jenkins/jenkins") sh -c "echo 'Hello, Jenkins!' && exit 0"
    - name: Stop Jenkins container
      run: docker stop $(docker ps -q --filter
"ancestor=jenkins/jenkins")
```

- **Test and Deploy the Project**
  Testing and continuous deployment using Github Actions (handled by another yml file). Once testing is complete, the team can deploy the project to production.

Project Deliverables

- May 4, 2023: Community bonding begins.
  I will make sure to interact with the Jenkins community and draft a proposed timeline for the project. The Dev environment will include essential Dev tools (like Git, JDK, IDE), Docker, Docker-Compose, and Kubernetes.

- May 29 (Coding Begins)
  Generate Docker instances, test them, and generate the corresponding docker-compose files.
- July 10 - July 14 (Standing Coding Period Evaluation)
  Docker instances would have been run, tested, and evaluated by this period for documentation generation.
- Aug 28 - Sept 4 (Final Evaluation for Standard Coding Period)
  An automated test framework will have been generated that ensures the correct functioning of the containers and alert the appropriate teams in case any container fails.

- Sept 5 (Initial Results Announced)
  The project will be finalized with community suggestions for the containers and documentation.
- Post GSOC - What will you do with what you have accomplished?
  Will make sure to have continuous ties with the Jenkins community to gain knowledge and provide insights wherever possible.

Proposed Schedule

March 20 - April 4, 2023 (Application Period)

Apply for the application to Jenkins community.

April 5 - May 3 (Acceptance Waiting Period)

Wait for acceptance from Jenkins.

May 4 - May 28 (Community Bonding Period)

I will make sure to interact with the Jenkins community and draft a proposed timeline for the project. The Dev environment will include basic Dev tools (like Git, JDK, IDE), Docker, Docker-Compose, and Kubernetes.

May 29 - July 14 (Standard Coding Period)

Generate Docker instances, test them, and generate the corresponding docker-compose files.
I will also ensure that I am in constant touch with the community.

July 14 - August 28 (Work Period for Standard Route)

By this period, Docker instances would have been run, tested, and evaluated for documentation generation.

Continued Involvement

Initially, I would start as a community member, gaining knowledge from various forums and contributing my knowledge wherever possible to ensure full-fledged discussion on topics.

Conflict of Interests or Commitment(s)

- May 1-May 7 (Exams)
- July 1 – July 30 (Internship Season)

Major Challenges Foreseen

Developing this project could also face several challenges, including:

- Technical Skills: Developing a project like this requires understanding Docker, Jenkins, and other related technologies. The team must understand software development principles, infrastructure management, and testing methodologies well.

- Resource Management: The team must be mindful of resource utilization while running the Docker instances. They need to be aware of the system requirements, such as memory and CPU usage, and optimize the Docker images to reduce their size and resource consumption.

- Scalability: The project needs to be scalable to support an increasing number of users and types of Jenkins instances. The team needs to design the architecture, such as load balancing and clustering, to ensure the system can handle increased traffic.

- Collaboration: Collaboration between the development and documentation teams is essential to ensure that the documentation is up-to-date, accurate, and relevant to the project's goals.

- Continuous Integration/Continuous Deployment (CI/CD): The project's continuous integration and deployment pipeline must be carefully designed to ensure that the Docker images are automatically built, tested, and deployed to production. The pipeline should also be flexible enough to handle frequent changes in the project.

References

Docker Documentation
Kubernetes Documentation
Jenkins Documentation
Jenkins Community

Relevant Background Experience

Worked with various docker instances while completing the course “Docker Training” by KodeKloud.
Have already worked with various OSes due to University Courses.
Contributed over 8,000 lines of code in Github’s HacktoberFest 2022 GITHUB

Personal

Hello, my name is Arnav Gupta,
and I am a self-taught developer and programmer with a strong passion for innovation, problem-solving, and software development. As a primary user of Java, I regularly participate in various development competitions and enjoy contributing to the open-source community. I'm also an avid sports enthusiast who loves to play, and I enjoy traveling to new places and having adventures.

I am pursuing a Bachelor's Degree at the prestigious Indraprastha Institute of Information Technology Delhi (IIITD). In my free time, I'm exploring the world of Competitive Coding and honing my skills to become an even better coder.
My accomplishments include achieving a 1606 rating on CodeChef, a 1067 rating on

Codeforces, and a 5-star rating on HackerRank (Java).

I have already contributed to various open-source projects in Github's HacktoberFest 2022, and this is a step for contributing towards more.

Experience

Free Software Experience/Contributions (optional):

- Worked on the open-source project Github/Sherlock-project LINK
- Contributed over 8,000 lines of code in Github's HacktoberFest 2022 GITHUB

Language Skill Set

Core Languages:

Java, Python, C, C++, SQL, MATLAB, Kotlin, HTML/CSS

Tools and Frameworks:

Gradle, Maven, Git/GitHub, libGDX, Docker, Docker Swarm, Kubernetes, Nginx, Android JVM, Linux

Scripting, Cloud, and Database:

Selenium, Beautiful Soup, Bash Scripting, Google Cloud, AWS, Firebase, MongoDB, SQLite

Reference Links and Web URLs (optional):

GitHub: https://github.com/arnavgupta2003

LinkedIn: https://www.linkedin.com/in/arnavgupta-/

Dev: https://dev.to/arnavgupta2003

LeetCode: https://leetcode.com/extinct_arnav/