**Wizard's Almanac’s Jenkin CICD pipeline user manual**

1. **Advantages of building a Jenkin CICD pipeline for Wizard's Almanac application**

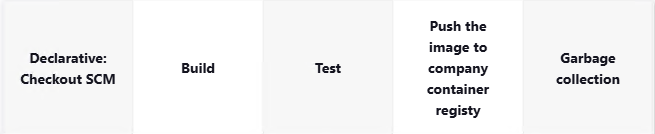
A CICD pipeline helps developers and companies the followings:

* Faster Development and Deployment: Automating repetitive tasks like testing and deployment, CI/CD accelerates the development cycle, enabling teams to release new features, bug fixes, and updates more quickly and frequently.
* Improved Code Quality: Continuous Integration ensures that new code is regularly merged into the main codebase and automatically tested for issues, which helps catch bugs early. This leads to higher-quality software and fewer issues during production.
* Automated Testing: With automated tests integrated into the CI/CD pipeline, developers can quickly verify that new code does not break existing functionality. This reduces the risk of introducing errors and ensures a more stable product.
* Reduced Manual Errors: Automating the build, test, and deployment processes minimizes the risk of human error. This ensures that releases are more predictable and stable.
* Consistent Environments: CI/CD ensures that code is tested and deployed in the same environment it will run in production, reducing issues related to environment differences (e.g., "it works on my machine" problems).\

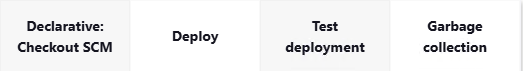
1. **How this Wizard's Almanac’s Jenkin CICD pipeline works**

The Wizard's Almanac’s Jenkin CICD pipeline is divided into two parts: The CI pipeline and the CD pipeline.

* CI pipeline: Used for continuous integration. The input of this pipeline is the source code from git repo and the output is a complete image which has been tested and pushed to company container registry. The CI pipeline is configured by the **jenkin-CI** file in the git repo.
  + There are five stages in CI pipeline: Checkout SCM (Check and pull source code from git repo) > Build (build image with docker file) > Test (run container and test the app) > Push the image to company container registry (publish image) > Garbage collection (remove all garbage left behind)



* CD pipeline: Used for continuous delivery. The input of this pipeline is the built image from the CI pipeline and the output is the deployment of the image onto an environment. The deployment is setup by docker compose. This CD pipeline is only run after an approval is secured. The CD pipeline is configured by the **jenkin-CD** file in the git repo
  + There are 4 stages in the CD pipeline: Checkout SCM (Check and pull source code from git repo) > Deploy (deploy the image with docker compose) > Test deployment (test the deployment with public IP of the app) > Garbage collection (remove all garbage left behind)



1. **How to setup jenkin to run these pipelines**

In this documentation, jenkin is run using docker. There are two ways to achieve this:

* The short way: pull and run the official jenkin image

|  |
| --- |
| docker run \  --name jenkins-pipeline \  --restart=always \  --detach \  --publish 8080:8080 \  --publish 50000:50000 \  --volume jenkins-data:/var/jenkins\_home \  --volume jenkins-docker-certs:/certs/client:ro \  jenkins/jenkins:lts-jdk17 |

* The long way: build jenkin image with dockerfile and run the image. This allows more control over jenkin‘s container and application.
  + Create and name this as Dockerfile-jenkin:

|  |
| --- |
| # Example Jenkins Dockerfile, modify this to complete the challenge  FROM jenkins/jenkins:lts  USER root  RUN apt-get update && apt-get install -y lsb-release  RUN curl -fsSLo /usr/share/keyrings/docker-archive-keyring.asc \  https://download.docker.com/linux/debian/gpg  RUN echo "deb [arch=$(dpkg --print-architecture) \  signed-by=/usr/share/keyrings/docker-archive-keyring.asc] \  https://download.docker.com/linux/debian \  $(lsb\_release -cs) stable" > /etc/apt/sources.list.d/docker.list  RUN apt-get update && apt-get install -y docker-ce-cli  USER jenkins  RUN jenkins-plugin-cli --plugins "blueocean docker-workflow" |

* + Build the jenkin image with the following command:

|  |
| --- |
| docker build -t myjenkins:lts -f Dockerfile-jenkin . |

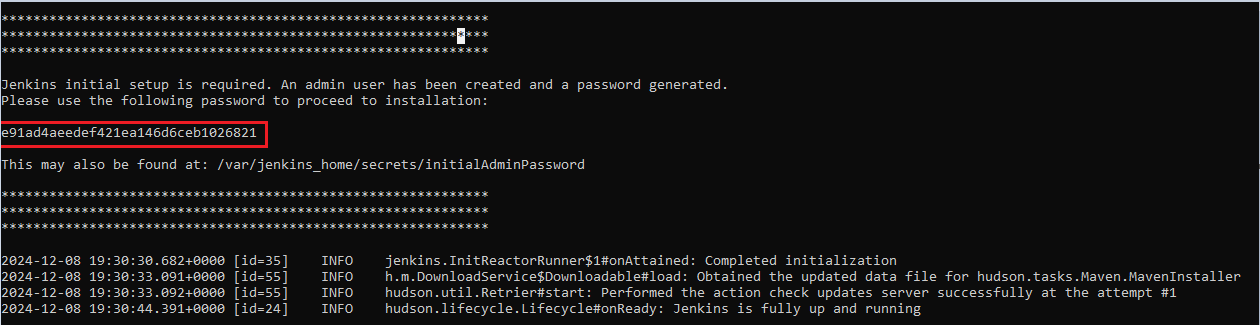
* + After build done, run the jenkin container with the following command:

|  |
| --- |
| docker run \  --name jenkins-pipeline \  --restart=always \  --detach \  --publish 8080:8080 \  --publish 50000:50000 \  --volume jenkins-data:/var/jenkins\_home \  --volume jenkins-docker-certs:/certs/client:ro \  myjenkins:lts |

Next, open web browser on [**http://localhost:8080/**](http://localhost:8080/)**.** Using this command to get jenkin’s initial password:

|  |
| --- |
| docker logs jenkins-pipeline |

Password should be found here:



Then, insert this password into jenkin console to start the setup process

1. **How to run CI pipeline**
2. **How to run CD pipeline**