

# Inject 2 Part 1 Walkthrough

---

This guide will be used to provide a walkthrough for Inject 2 Part 1 in an **insecure manner**. This should only be used as a last resort when the team could not fully complete the task in a reasonable amount of time.

## Table of Contents

---

- [Inject 2 Part 1 Walkthrough](#)
- [Table of Contents](#)
  - [Prompt](#)
  - [Requirements of the task](#)
  - [Infrastructure Diagram](#)
  - [Walkthrough](#)
    - [Custom Docker Images](#)
    - [Connecting Jenkins to the INSECURE Registry V2](#)
    - [Updating Jenkinsfile](#)
    - [Ansible Deployment](#)
    - [Internal Domain](#)

## Prompt

### **Inject # 2 - Part 1**

Zodu developers have been looking for a solution to help them in their efficiency of code development and deployment. And so, the IT team has reached out to you.

=====

Hi Blue Team,

We have heard concerns from the software development team that their efficiency rating has decreased due to the increased workload. And so, they are looking for ways to automate their development and deployment process. We would like to have our own internal Git server to house our code as we don't trust GitHub to store it safely and to keep our uploads encrypted.

We'd like a demonstration that will showcase the use of an internal git server used with some automation tool that can build, test, and deploy a custom made container onto two separate machines. This custom made container can be anything, we just want to see if it's possible to automate the software development and application deployment process.

We'd like to have the automation tool's domain to be accessible through the internal network.

Thanks.  
Zodu IT

=====

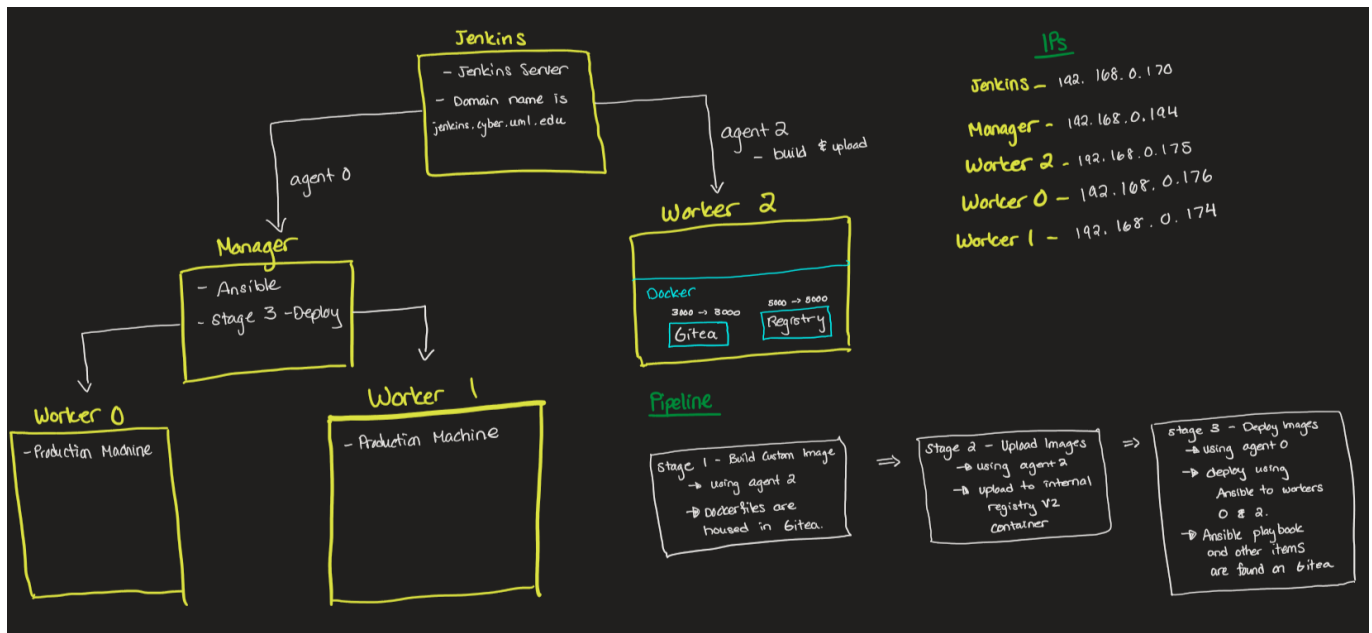
**Time Limit: 1 hour**

## **Requirements of the task**

There were various portions to this task that require collaboration between team members for the sake of the end goal. I will try to create a coherent set of steps for each portion. But for now, there needs to be a direction for each of the injects.

1. First off, we need an internal Git server on our network.
  1. Later stages would be to encrypt communications, but for now, keep it basic. Unless you have a complete guide that will allow you to make it secure from the get go.
2. "Automation tool that can build, test and deploy..." - This is textbook **Jenkins**.
  1. There are many steps that are needed in here, I'll explain them in the Jenkins section that is coming up.
3. "Custom made container..." - Sounds like a Dockerfile will need to be made. It can be stored on the Git server to be used by Jenkins during deployment.
4. "Two separate machines" - There's various ways that we can take this. We can do ssh commands into the different agent, we can have a machine that has Ansible installed to create the infrastructure on other machines, or we could do a docker-compose file.'
5. "Automation tool's domain to be accessible through the internal network". This can be done through a DNS entry that points to our Jenkins Server.

## **Infrastructure Diagram**



## Walkthrough

I will use premade guides to give you guidance on some of the portions for this task.

**Git Server Creation** - This guide was made by Rose which will create a git server ready for use.

**Jenkins Installation - EC2** - This guide was made by Chris. It goes through the process of creating EC2 machines on AWS and then installing a Jenkins Server onto one of them. Once that is done, then an agent will be registered onto the server for use in a Jenkinsfile. You can skip to the **Registering Agents** section within this guide to understand how to properly add a node into your server.

## Custom Docker Images

The prompt did not say how to customize these images. We can just write two custom images for deployment.

1. Image 1 - This is an image that contains various text editors such as vim, nano and emacs.
2. Image 2 - This is simply an image that contains vim.

Both of these have an entrypoint script that will keep the containers alive. These can be found in ../Custom-Images.

**Registry V2 Creation** - Use this with care. It's **INSECURE**. This is simply to get the registry up so Jenkins can push to it. There are more steps that I'll explain next.

## Connecting Jenkins to the INSECURE Registry V2

In the infrastructure diagram above, you can see that there are 2 deployment machines per the prompt. However, Docker tries to include some security when contacting registries that download images. You will notice that there is an error when you try to pull the image from our internal registry.

```
manager@manager:~$ docker pull 192.168.0.175:5000/registry-demo-uploaded
Using default tag: latest
Error response from daemon: Get "https://192.168.0.175:5000/v2/": http: server gave HTTP response to HTTPS client
manager@manager:~$
```

*Note: This pull was done from a completely different machine. This is to showcase this error. Localhost wouldn't have this problem.*

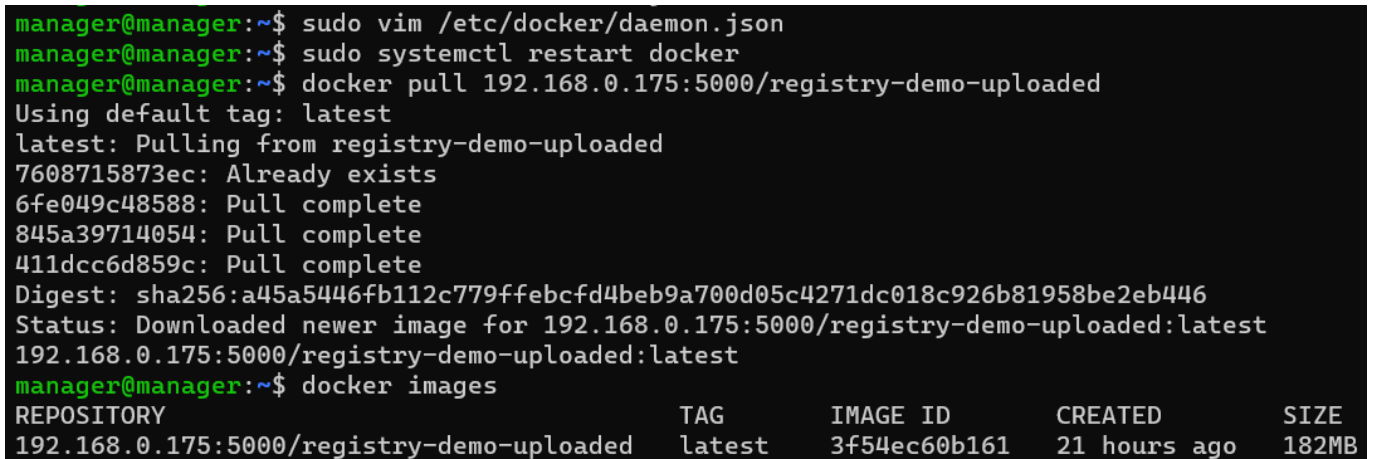
To fix this problem, we will need to add an extra setting in the Docker daemon file to allow this connection. This can be found in `/etc/docker/daemon.json`. You may need to create this file.

The file will look like:



```
manager@manager: ~  
"insecure-registries" : ["http://192.168.0.175:5000"]
```

Replace the IP with whatever you need. Now, you need to **restart the Docker service** for it to take effect.



```
manager@manager:~$ sudo vim /etc/docker/daemon.json  
manager@manager:~$ sudo systemctl restart docker  
manager@manager:~$ docker pull 192.168.0.175:5000/registry-demo-uploaded  
Using default tag: latest  
latest: Pulling from registry-demo-uploaded  
7608715873ec: Already exists  
6fe049c48588: Pull complete  
845a39714054: Pull complete  
411dcc6d859c: Pull complete  
Digest: sha256:a45a5446fb112c779ffebcf44beb9a700d05c4271dc018c926b81958be2eb446  
Status: Downloaded newer image for 192.168.0.175:5000/registry-demo-uploaded:latest  
192.168.0.175:5000/registry-demo-uploaded:latest  
manager@manager:~$ docker images  
REPOSITORY                                TAG      IMAGE ID      CREATED      SIZE  
192.168.0.175:5000/registry-demo-uploaded latest   3f54ec60b161   21 hours ago 182MB
```

Ideally you just put the certs that are needed. But, if you absolutely cannot, then this is how you will get the end product done in an insecure way.

Whoever needs to use this registry will need to have this fix implemented for them.

## Updating Jenkinsfile

Now, we need to try and automate the building stage on the agents now. This is not the final Jenkinsfile. But it's close to it.



```
stage('Build Custom Image - Agent 2'){
    agent {
        label 'agent2'
    }
    steps {
        sh '''
            docker images

            docker build custom-images/image1/ -t custom-image-1
            docker build custom-images/image2/ -t custom-image-2

            docker images
            ...
        '''
    }
}
stage('Upload Images - Agent 2'){
    agent {
        label 'agent2'
    }
    steps {
        sh '''
            docker tag custom-image-1 localhost:5000/custom-image-1-uploaded
            docker push localhost:5000/custom-image-1-uploaded

            docker tag custom-image-2 localhost:5000/custom-image-2-uploaded
            docker push localhost:5000/custom-image-2-uploaded

            docker images

            docker rmi custom-image-1
            docker rmi custom-image-2
            docker rmi localhost:5000/custom-image-1-uploaded
            docker rmi localhost:5000/custom-image-2-uploaded

            docker images

            docker pull localhost:5000/custom-image-1-uploaded
            docker pull localhost:5000/custom-image-2-uploaded

            docker images
        '''
    }
}
```

```
stage('TEMP - Agent 0 - Download remote images'){
    agent {
        label 'agent0'
    }
    steps {
```

```
sh '''
    docker images
    docker pull 192.168.0.175:5000/custom-image-1-uploaded
    docker pull 192.168.0.175:5000/custom-image-2-uploaded

    docker images
    docker run -d --name ci1 192.168.0.175:5000/custom-image-1-uploaded
    docker run -d --name ci2 192.168.0.175:5000/custom-image-2-uploaded

    docker ps -a
    ...
}
}
```

These two images showcase a Jenkinsfile that will build the image on agent2 and then push it to the registry contained on agent2's machine. Finally, it will try to pull the image from our internal registry and spin up a container using those images.

## Ansible Deployment

Next, this is where the deployment onto the other machines will be done. This step could be done in many ways, but to showcase the ease of integrating the automation tools together to do greater things.

And so, I wrote an ansible script to deploy these machines on different hosts.

The files look as follows:

### Inventory File

```
[deployment-machines]
192.168.0.174 ansible_ssh_private_key_file=~/.ssh/ansible_id_rsa
192.168.0.176 ansible_ssh_private_key_file=~/.ssh/ansible_id_rsa
```

### Ansible Playbook

```

- name: Deploy image 1 to machines
  hosts: deployment-machines
  tasks:
    - name: Check if hosts are up
      ansible.builtin.ping:

    - name: Create image 1 container
      community.docker.docker_container:
        name: im1
        hostname: im1
        image: 192.168.0.175:5000/custom-image-1-uploaded
        restart: true

    - name: Create image 2 container
      community.docker.docker_container:
        name: im2
        hostname: im2
        image: 192.168.0.175:5000/custom-image-2-uploaded
        restart: true

```

The playbook simply just spins up two docker containers *not* on their own network. To follow what was asked, this is all this is needed as proof.

We can confirm that the containers were made by looking on the deployment machines. I only show one in the guide, but in a write-up you want both. And you want the command line to show the hostname.

|              |  |                          |             |            |
|--------------|--|--------------------------|-------------|------------|
| 161c7e2abeb7 | 192.168.0.175:5000/custom-image-2-uploaded | "/bin/sh /entrypoint..." | 6 hours ago | Up 6 hours |
|              | im2  |                          |             |            |
| 00613367013e | 192.168.0.175:5000/custom-image-1-uploaded | "/bin/sh /entrypoint..." | 6 hours ago | Up 6 hours |
|              | im1  |                          |             |            |

Upon further inspection, you can simply type **vim**, **emacs** and **nano** on each machine and confirm that the expected editors from the Dockerfile are present and nothing else.

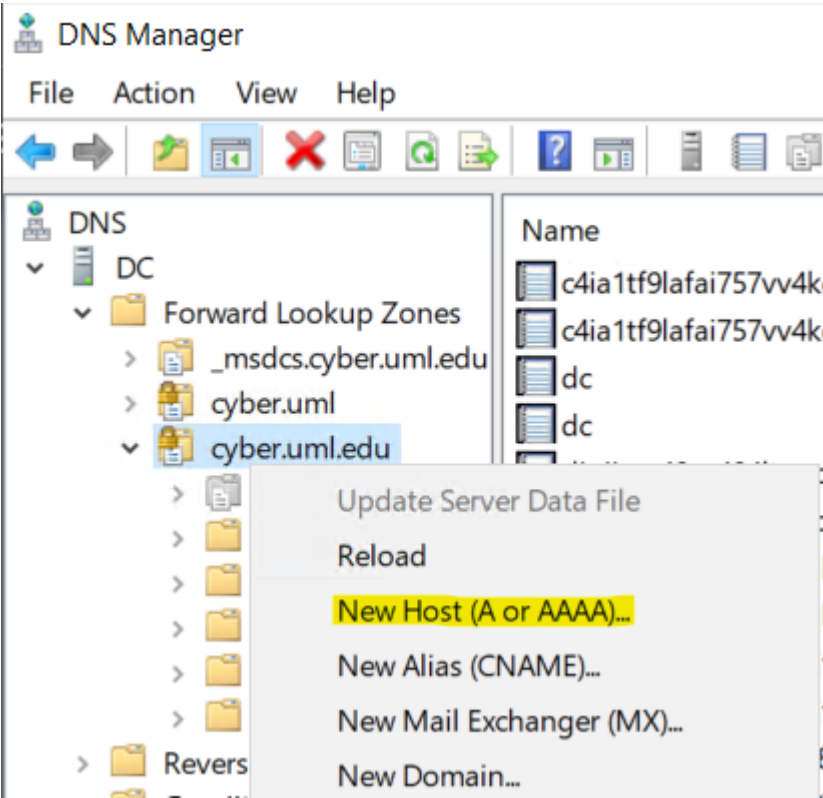
## Internal Domain

The final part of the inject is to have the "automation tool" accessible through a Domain(Jenkins).

*This guide assumes that you already have a working DC DNS instance with a domain already installed. I will simply be showing you how to add the domain to the DNS.*

1. First thing to do is to add to add a new host into the overall domain.





2. Now, we just fill in the dialog box with the information that we desire.

New Host

Name (uses parent domain name if blank):

jenkins

Fully qualified domain name (FQDN):

jenkins.cyber.uml.edu.

IP address:

192.168.0.170

☐ Create associated pointer (PTR) record

☐ Allow any authenticated user to update DNS records with the same owner name

Add Host

Cancel

3. Now, we have a complete entry inside the forward lookup zone.

DNS Manager

File Action View Help

DNS

DC

Forward Lookup Zones

\_msdcs.cyber.uml.edu

cyber.uml

cyber.uml.edu

\_msdcs

\_sites

\_tcp

\_udp

DomainDnsZones

ForestDnsZones

Reverse Lookup Zones


Conditional Forwarders

| Name                           | Type                  | Data                            | Timestamp |
|--------------------------------|-----------------------|---------------------------------|-----------|
| c4ia1tf9lafai757vv4kd209vfn... | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| c4ia1tf9lafai757vv4kd209vfn... | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| dc                             | Host (A)              | 192.168.0.191                   | static    |
| dc                             | RR Signature (RRSIG)  | [A][Inception(UTC): 2/11/202... | static    |
| djoiicce49as484ltggqfgnt8o...  | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| djoiicce49as484ltggqfgnt8o...  | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| e9dv3a83fda6k90hub285e30...    | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| e9dv3a83fda6k90hub285e30...    | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| fsnrclpvnduvejbikgjf4tjhd38... | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| fsnrclpvnduvejbikgjf4tjhd38... | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| ik4hsn0pt7dkn58jm8pl4oan...    | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| ik4hsn0pt7dkn58jm8pl4oan...    | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| ili9goc5m14r9fi9hedrd2lv1m...  | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| ili9goc5m14r9fi9hedrd2lv1m...  | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| jbv9epfkfqran9fts7nof2do...    | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |
| jbv9epfkfqran9fts7nof2do...    | Next Secure 3 (NSEC3) | [SHA-1][NO Opt-Out][50][4...    | static    |
| jenkins                        | Host (A)              | 192.168.0.170                   | static    |
| jenkins                        | RR Signature (RRSIG)  | [A][Inception(UTC): 2/10/202... | static    |
| jl8ptku7omspccraj5du9eba1...   | RR Signature (RRSIG)  | [NSEC3][Inception(UTC): 2/1...  | static    |

4. If we go into a web browser that has our internal DNS as a DNS server and use "jenkins.cyber.uml.edu" and using the default Jenkins port 8080, we will be able to access it.

Sign in [Jenkins]

jenkins.cyber.uml.edu:8080/login?from=%2F



Welcome to Jenkins!

Username

Password

☐ Keep me signed in

Sign in