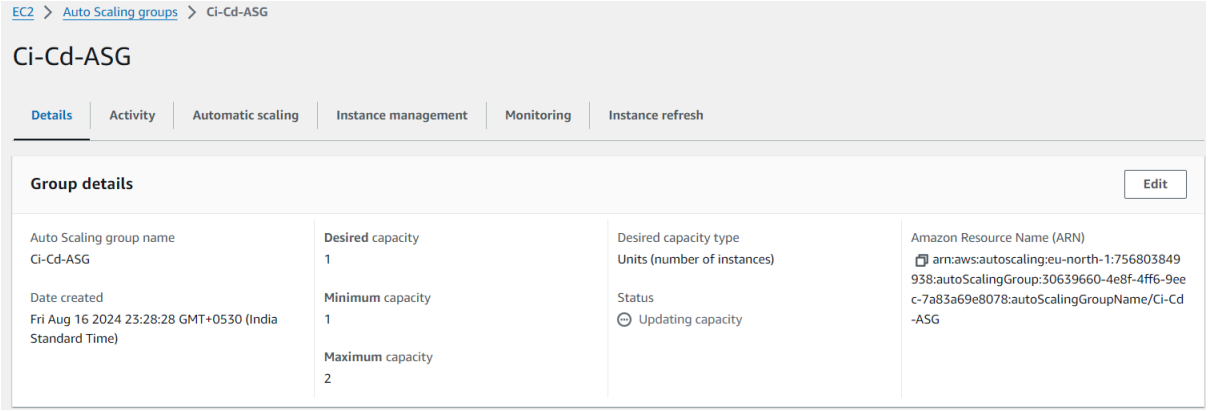
Jenkins Pipeline for Auto Deployment in EC2 (Autoscaling)

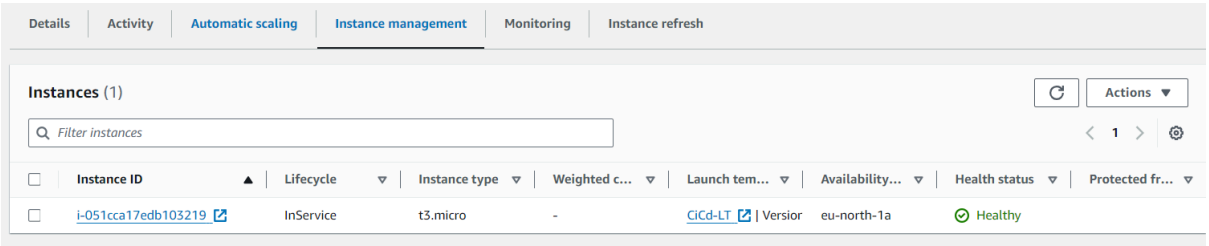
Project Overview

The goal of this project is to automate the deployment of code to new instances within an AWS Auto Scaling group using Jenkins. The system is designed to handle new instances that are automatically launched by Auto Scaling and deploy the latest code updates from a GitHub repository.

1. AWS Auto Scaling

To automatically manage and scale EC2 instances based on demand. Auto Scaling Group manages the number of EC2 instances in a specified range. It ensures that the desired number of instances are running at all times, and it can launch or terminate instances based on defined policies.



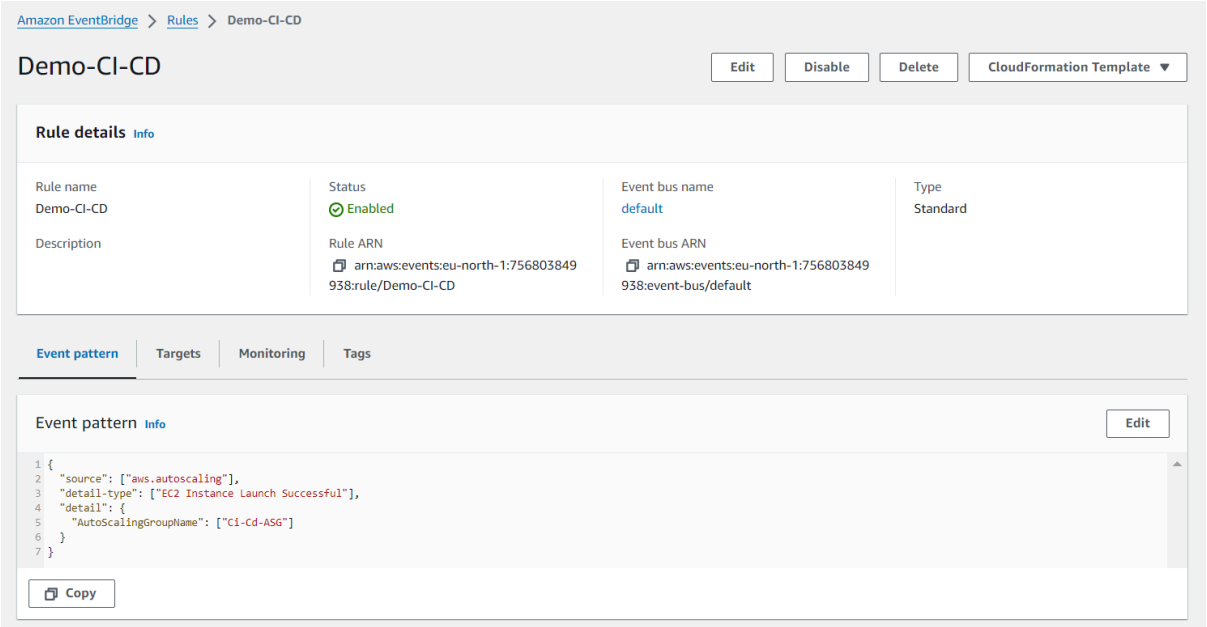


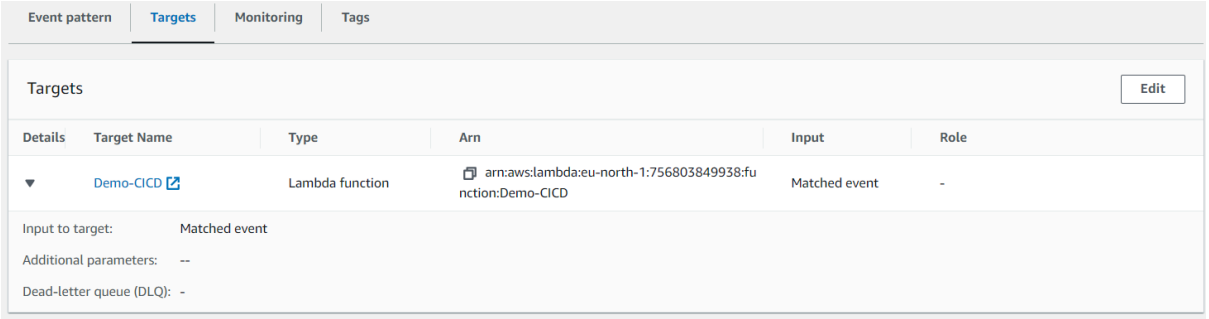
2. Triggering the Jenkins Pipeline

To initiate a trigger to Jenkins pipeline whenever a new instance is launched in the AutoScaling group.

EventBridge Rule:

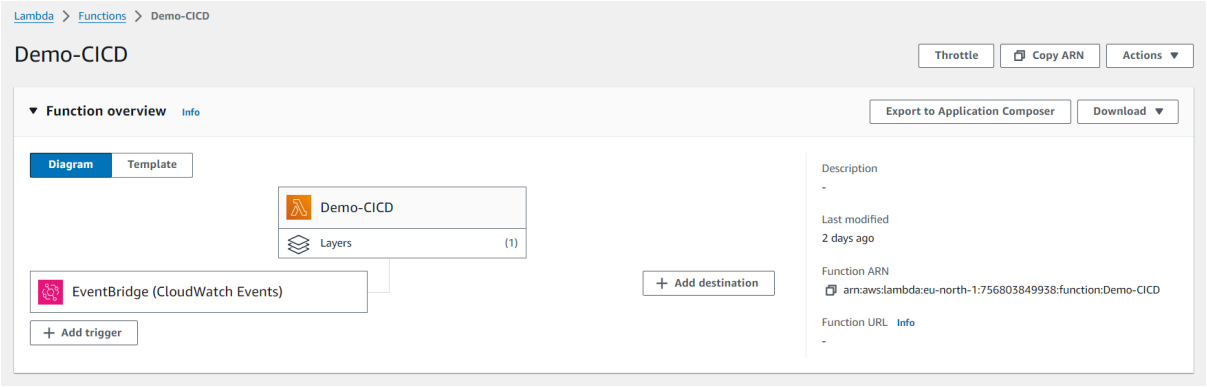
AWS EventBridge is configured to detect when a new instance is launched in the Auto Scaling group. The Event pattern in the rule is configured to trigger the target (lambda) if a new instance is launched in the Auto Scaling group.

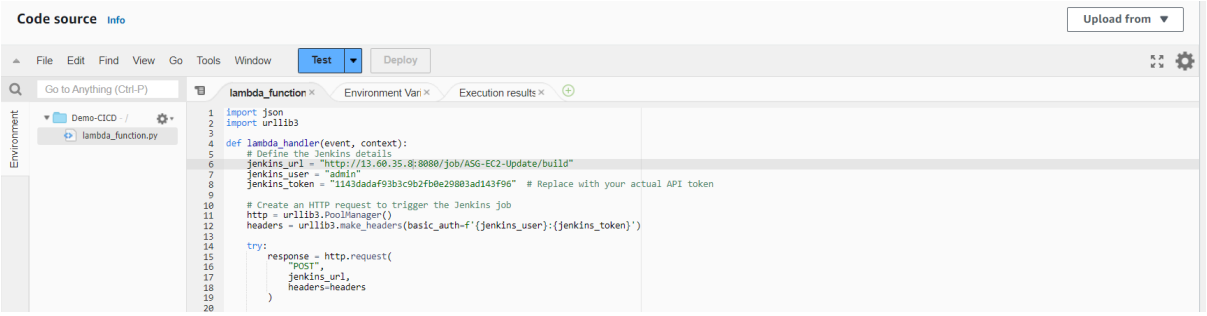




Lambda Function:

A Lambda function is triggered by the EventBridge rule. This function then triggers the Jenkins pipeline using Http post request with jenkins job url, jenkins username and jenkins api token.

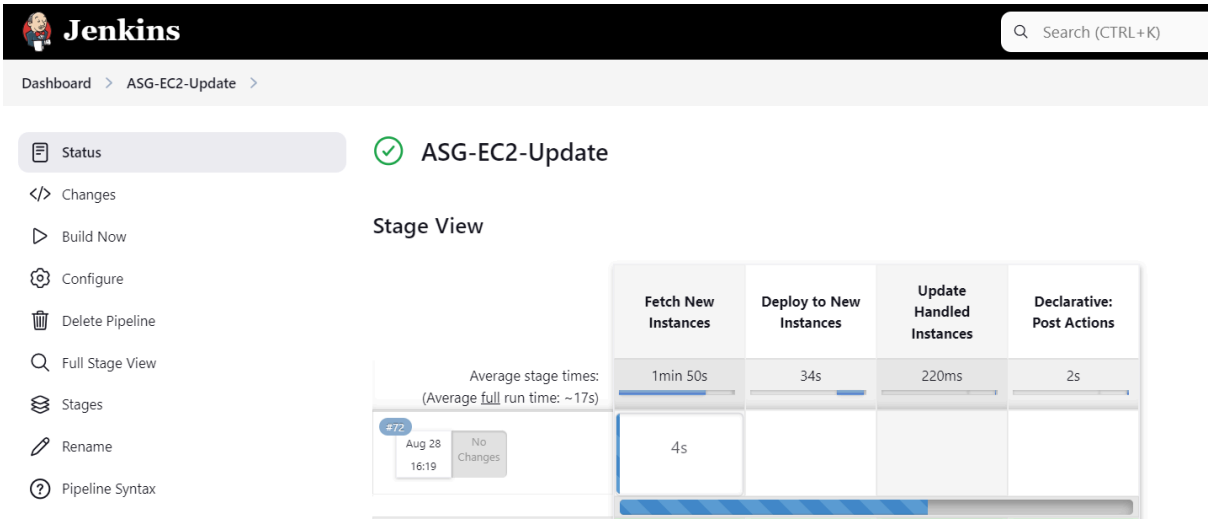


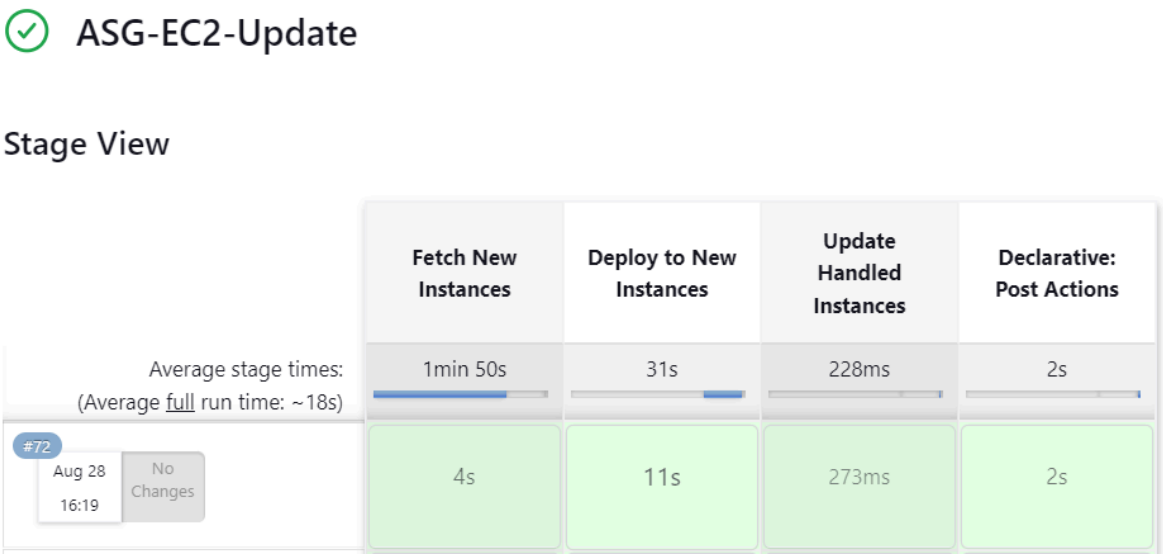


3. Jenkins Pipeline Execution

To deploy the latest code to newly launched instances in Autoscaling Group automatically.

Pipeline Trigger: The Jenkins pipeline is triggered by the Lambda function and Deploy to the newly launched instance in the autoscaling group.

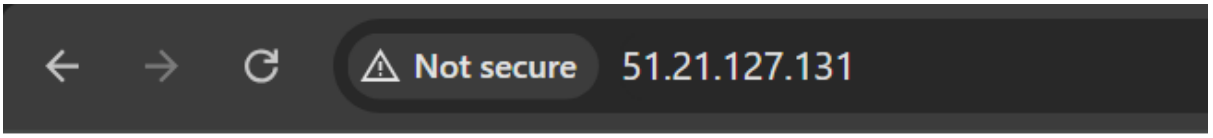




Code Deployment:

The pipeline fetches the latest code from a GitHub repository and deploys it to the new instance. Now the new instance in the Autoscaling group was deployed successfully.

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability ...	Public IPv4 address	Elastic IP
ASG - 1	i-051cca17edb103219	Running	t3.micro	3/3 checks passed	View alarms	eu-north-1a	51.21.127.131	–



My Server IP address is: 172.31.26.38

Deployment Process in the Jenkins Pipeline

1. Fetching New Instances: Identify the new instances that require deployment.

- **Instance Identification**: Once the pipeline is triggered, it first fetches the list of all instances in the Auto Scaling group that are in the "InService" state.
- **Determining New Instances**: The pipeline reads a file that tracks instances that have already been handled. It compares the list of current instances against this file to determine which instances are new and need deployment.

2. Deploying Code to New Instances: Deploy the latest code from the GitHub repository to the identified new instances.

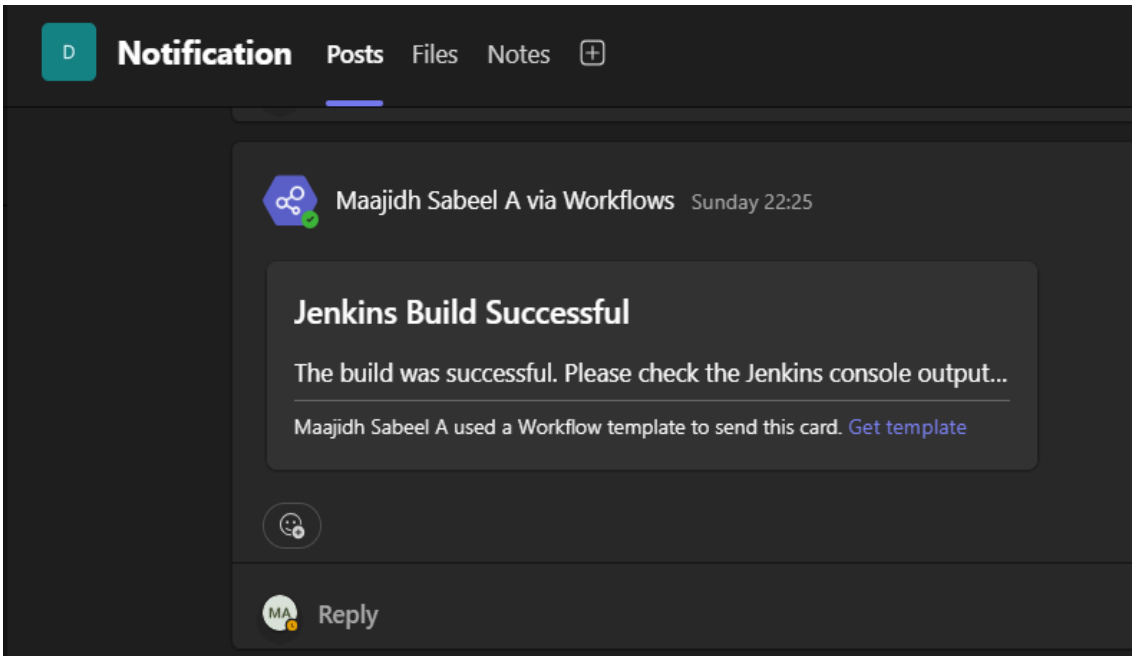
- **Clone Repository**: The pipeline clones the latest code from the specified GitHub repository. This ensures that the deployment always uses the most up-to-date code.
- **SSH Connection Setup**: The pipeline connects to each new instance via SSH. This requires:
    - **SSH Key**: The private key stored securely in Jenkins is used to authenticate and connect to the new EC2 instances.
    - **SSH Options**: Options like StrictHostKeyChecking=no are used to bypass host key verification, simplifying connections to fresh instances.
- **Deploy Commands Execution**: After establishing an SSH connection, the pipeline executes a series of commands on the new instance to deploy the code:
    - **Install Required Software**: Ensure necessary software (e.g., Git, application dependencies) is installed.
    - **Code Deployment**: Clone the GitHub repository or copy files to the appropriate directories on the instance.
    - **Service Restart**: Restart any services (e.g., web servers, application servers) to apply the new code.
- **Error Handling and Retries**: If an SSH connection fails or any command errors out, the pipeline retries the connection multiple times to ensure robustness. This is crucial for handling network instability or temporary instance unavailability.

3. Updating Handled Instances: Keep track of all instances that have been processed to avoid redeploying to the same instances in future runs.

- **Handled Instances**: After successfully deploying to an instance, the pipeline adds its IP address to the handled_instances.txt file. This file is then used in future pipeline runs to skip instances that have already been updated.

4. **Notifications**: In the post action of the pipeline the build status of the pipeline will be notified.

- **Success Notification**: If the deployment is successful for all instances, a success message is sent to a configured Microsoft Teams channel using a webhook.
- **Failure Notification**: If the deployment fails for any instance, a failure message is sent, allowing for quick troubleshooting and response.



Summary :

The Jenkins pipeline is designed to automate the entire deployment process for new instances launched by AWS Auto Scaling. It dynamically identifies new instances, deploys the latest code, and keeps track of handled instances, ensuring a smooth and efficient deployment process without manual intervention. Notifications provide real-time updates on deployment status, allowing teams to respond promptly to any issues.