**IOS MCN v0.1.0 Agartala Release**

**Developer Guide: SD-Core v0.1**

1. Introduction

This guide mentions the options to work with IOS-MCN github repository for the software development for SD-Core.

1. Purpose and Audience

This document is intended only for the core developers for the 5G platform.

1. How to build from source code

Clone the repository using the command

*git clone --recursive https://github.com/ios-mcn-core/ios-mcn-core.git*

*For private repo: The git command will ask for username and password. Provide the github account username. The password requires, provide github ‘Personal access tokens’ as password. For generating Personal access tokens, go to the Github account -> Settings -> Developer Settings. Go to personal access tokens and generate new token. Complete the authentication and issue the token as git hub password.*

* 1. Continuous Integration
* Create github and docker account for CI/CD and fork the repository.
* Go to **Actions** tab.
* Click on “I understand my workflows, go ahead and enable them”

# Creating Self Hosted Runner

* Go to **Settings** tab.
* In the left sidebar, click on **Actions**, and then click **Runners**.
* Click **New self-hosted**runner.
* Select the operating system image and architecture of the self-hosted runner machine.
* There will be instructions showing how to download the runner application and install it on the self-hosted runner machine.

# Creating Workflow

* Create a new workflow or use existing one.
* For creating new workflow, follow the [*Quickstart for GitHub Actions*](https://docs.github.com/en/actions/quickstart#creating-your-first-workflow)*.*
* Define the name of workflow.
* Specify the events that trigger the workflow.

# Using Jobs

* Setup jobs **for building, unit test, creating Docker image and pushing Docker image** to Docker Hub (Docker build and Docker push commands will be present in Makefile of all the network functions).
* Follow the steps in [Using jobs in a workflow](https://docs.github.com/en/actions/using-jobs/using-jobs-in-a-workflow#overview) for getting an idea on creating jobs.

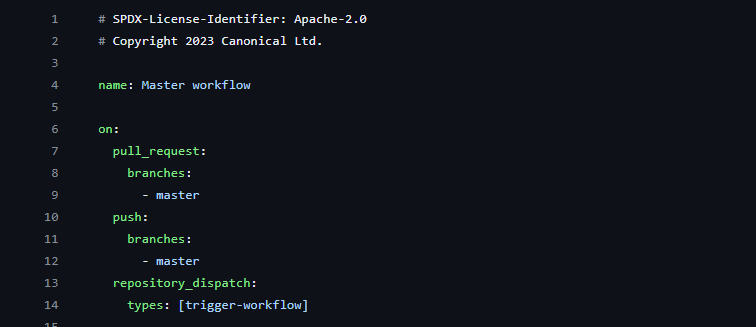


Figure 1: Sample workflow file

* For building the network function image, refer the Makefile located in the root of the repository and try to invoke the Docker build command as a GitHub actions job.

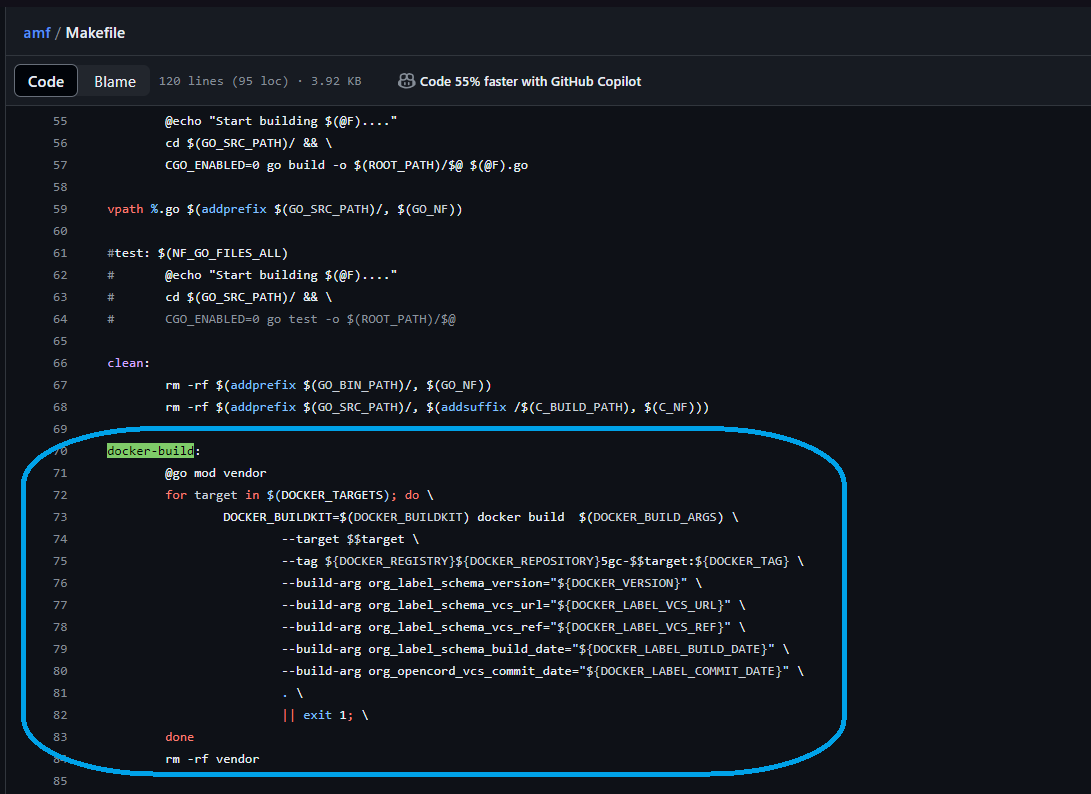


Figure 2: docker-build command present in Makefile

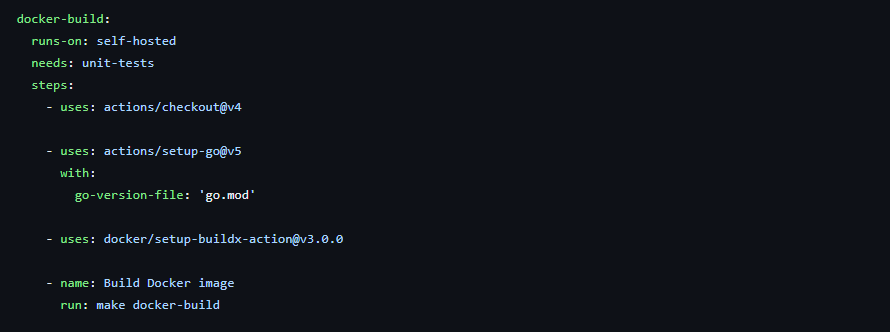


Figure 3: Invoke make docker-build inside a job

* + For pushing the job to DockerHub, refer the Makefile and try to invoke the Docker push command as a GitHub actions job. Don’t forget to set environments variables mentioned in the Docker push command.

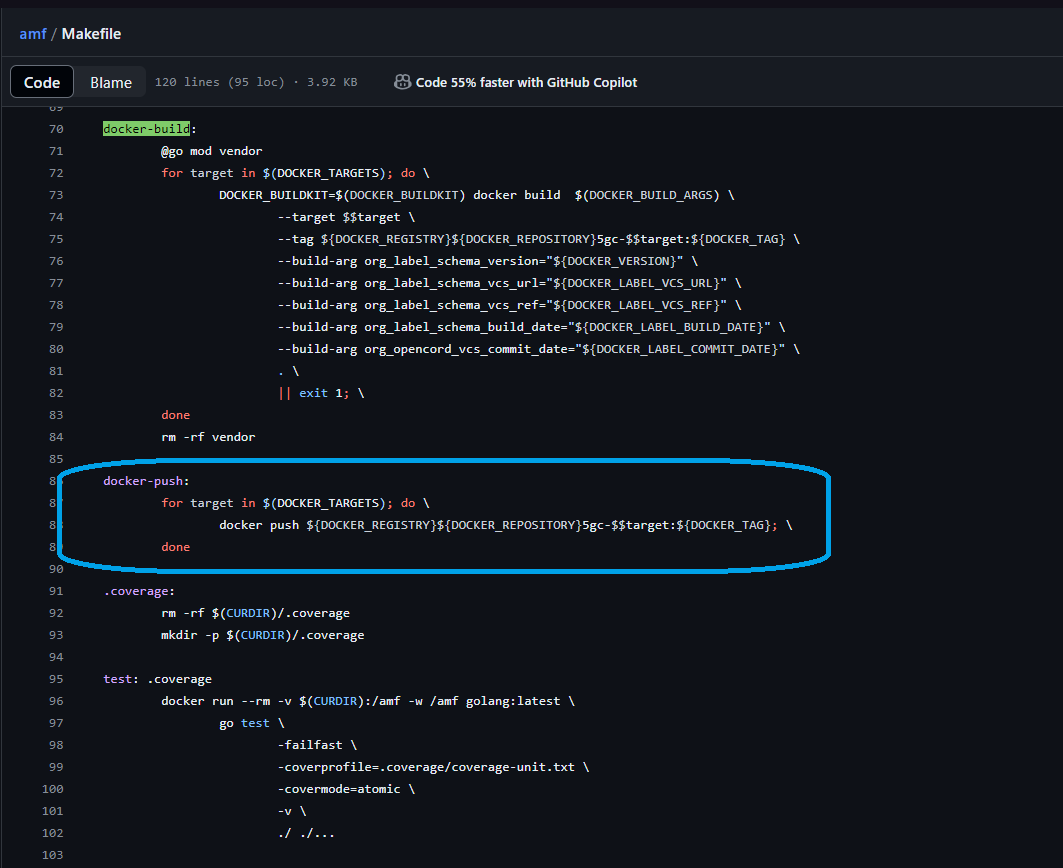


Figure 4: docker-push command present in Makefile

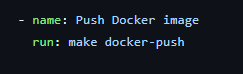


Figure 5: Invoke make docker-push inside a job

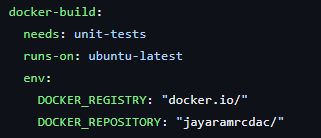


Figure 6: Registry and repository details

* + Alternative method of pushing Docker image without using Makefile command is to use Docker image already built and push it to the DockerHub manually using Docker commands.

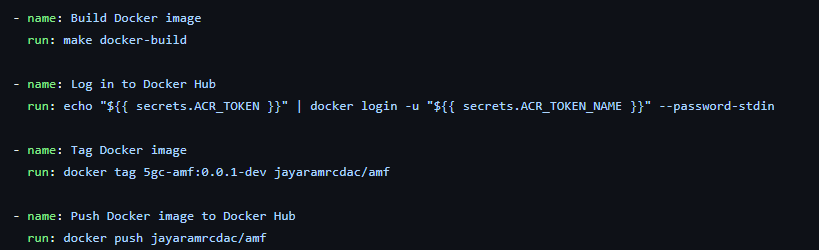


Figure 7: Tag and push already build image to DockerHub

* + Try to run the jobs in sequential order using ‘needs’ keyword.

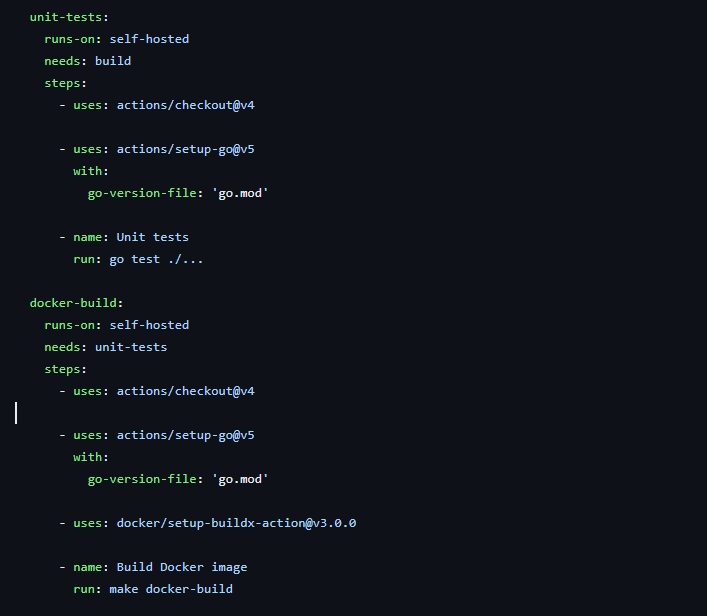


Figure 8: Usage of needs

* Use [JayaramRCDAC/amf](https://github.com/JayaramRCDAC/amf/blob/master/.github/workflows/master.yml) for reference.
* To add Docker credentials as **secrets**, follow [Using secrets in GitHub Actions](https://docs.github.com/en/actions/security-guides/using-secrets-in-github-actions#about-secrets)



Figure 9: Setting secrets

* + GitHub provides a **marketplace** where pre-built actions are present to use in the workflows.

# Running Workflow

* Committing the workflow file to a branch in the repository triggers the push event and runs the workflow.

# Monitoring Workflow

* Monitor the progress of workflows in the **Actions** tab of the repository.
* Check the status of each workflow run, view logs, and debug any issues that arise.

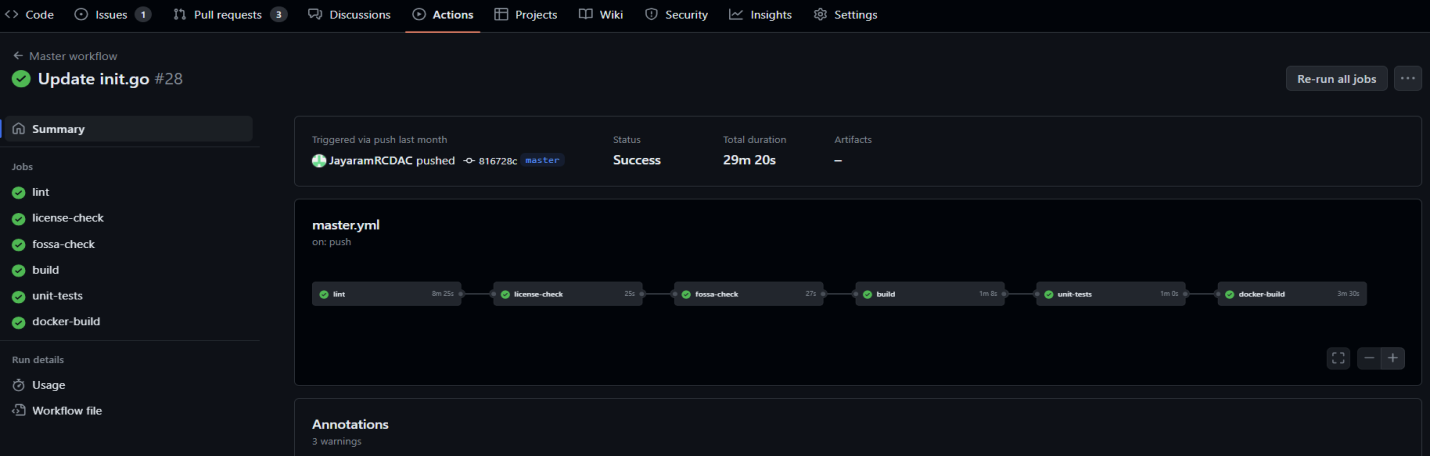


Figure 10: Sample workflow pipeline

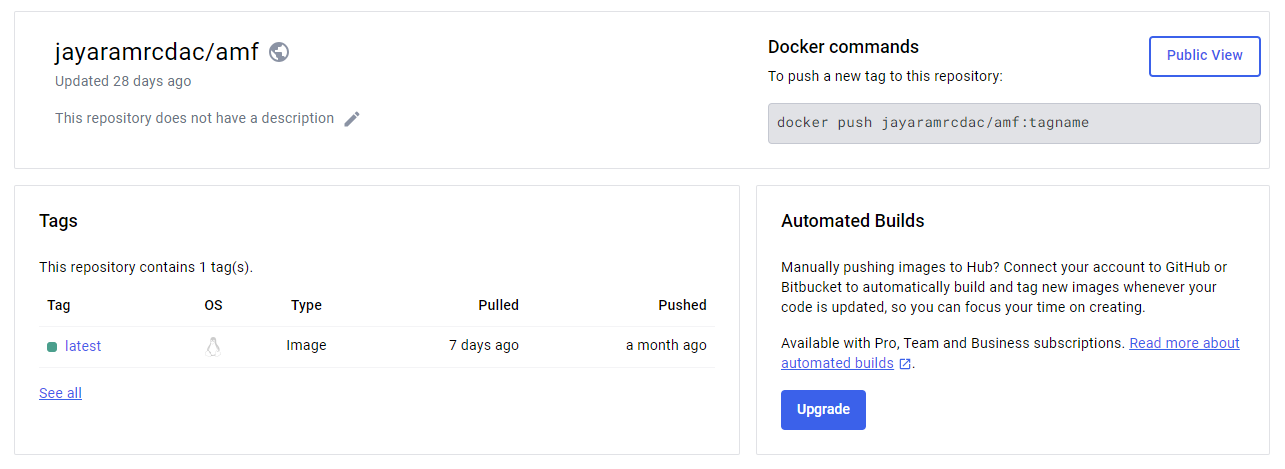


Figure 11: Image pushed to DockerHub

# Integration with IDE

* Download and setup **VSCode** IDE.
* Clone a repository using command palette (> git clone).
* Install **GitHub Actions** extension.
* Sign in with the GitHub account and when prompted allow GitHub Actions access to the GitHub account.
* Push some changes to master (refer GitHub guide document) & the triggered workflow will be visible at GitHub actions side panel.

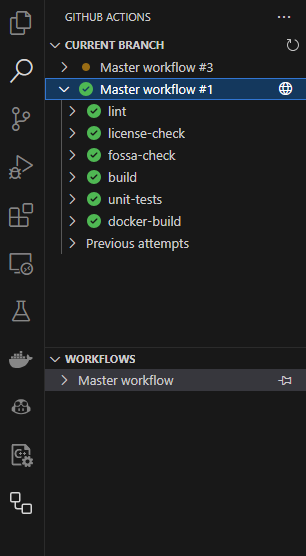


Figure 12: Workflow log inside VSCode

* 1. Deployment
  + Open 5g-control-plane folder in sdcore and edit values.yaml file
  + Change image names in tags to ios-mcn
  + Change 5g-control-plane dependency repository location to above extracted file location
  + Run the following commands

*make reset-5g-test*

*make 5g-test*

* + Check the pod status using the command

*kubectl get pods -n ios-mcn*

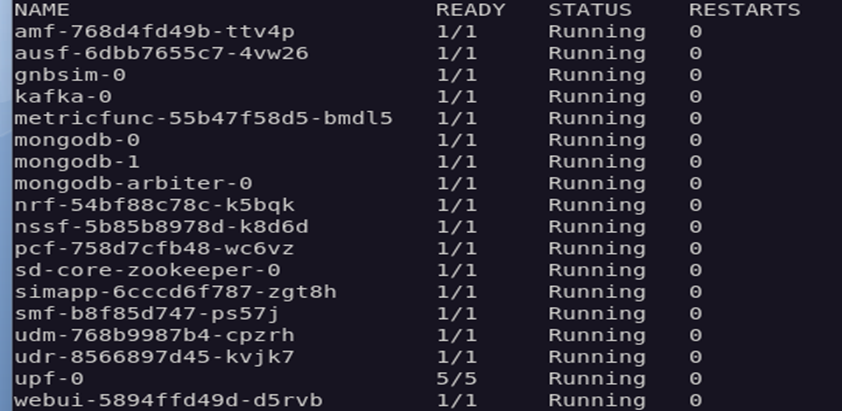


Figure 13: pods status

1. Related Artifacts & links

Give all related documents list and links (table is ok)

|  |  |  |
| --- | --- | --- |
| **Document Name** | **Purpose** | **Link** |
| User Guide |  |  |
| Installation Guide |  |  |
| Developer Guide |  |  |
| Troubleshooting guide |  |  |