**STEPS TO IMPLEMENT THIS PROJECTS.**

**SET UP A KUBENETES CLUSTER**

For setting up Kubernetes cluster, we will do the following:

* Set up infrastructure (provision nodes, install container runtime)
* Install kubeadm and dependencies on initialization machine
* Initialize cluster on master node with kubeadm
* Join worker nodes to the cluster with kubeadm
* Set up networking (CNI plugin, load balancer)
* Deploy a CNI plugin for networking

Refer: <https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/>

**SETUP JENKINS ON KUBERNETES**

For setting up a Jenkins Cluster on Kubernetes, we will do the following:

* Create a Namespace
* Create a service account with Kubernetes admin permissions.
* Create local persistent volume for persistent Jenkins data on Pod restarts.
* Create a deployment YAML and deploy it.
* Create a service YAML and deploy it.

Refer [https://devopscube.com/setup-jenkins-on-kubernetes-cluster/](https://devopscube.com/setup-jenkins-on-kubernetes-cluster/%20) for step by step process to use these manifests.

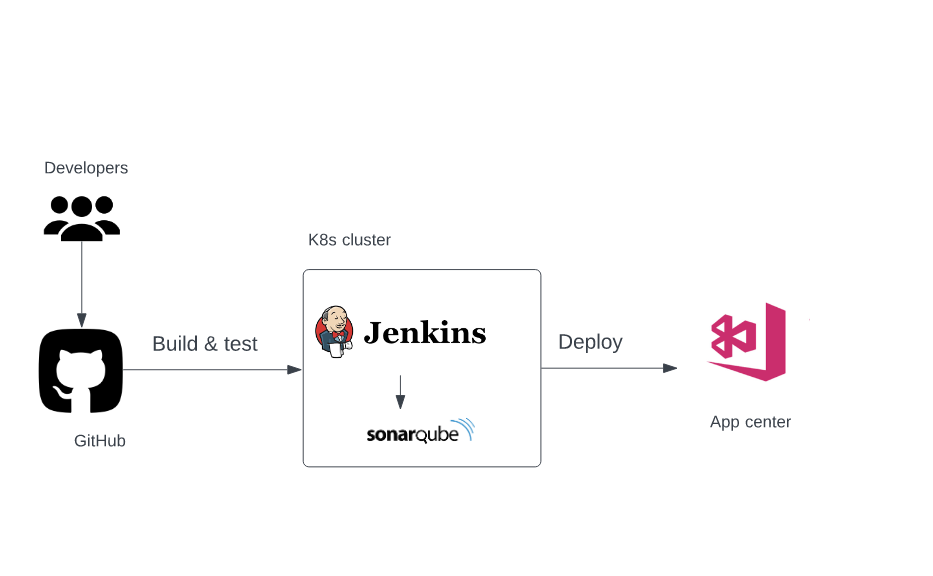
**SETUP SONARQUBE ON KUBERNETES CLUSTER**

Steps involved in deploying SonarQube on a Kubernetes cluster:

* Create a Kubernetes namespace for SonarQube.
* Create a PV and PVC for SonarQube's data.
* Create a secret for SonarQube's database password.
* Create a ConfigMap for SonarQube's configuration.
* Deploy a PostgreSQL database.
* Deploy SonarQube using a deployment and service, with PV, PVC, secret, and ConfigMap.
* Expose the SonarQube service with Ingress or LoadBalancer.

Refer: <https://docs.sonarqube.org/9.8/setup-and-upgrade/deploy-on-kubernetes/deploy-a-sonarqube-cluster-on-kubernetes/>

**ARCHICTECTURE**

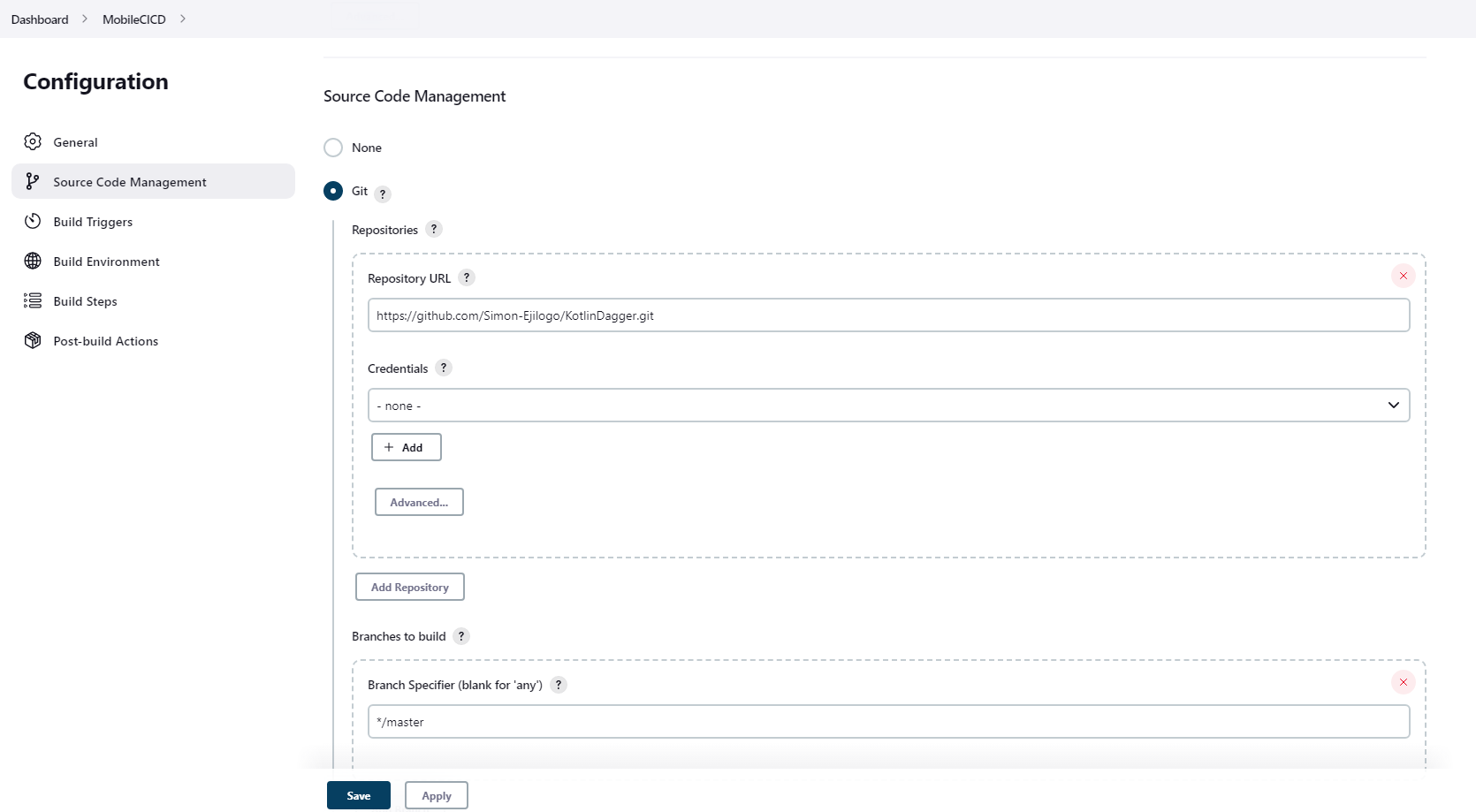


**CONNECT JENKINS TO GITHUB FOR CICD**

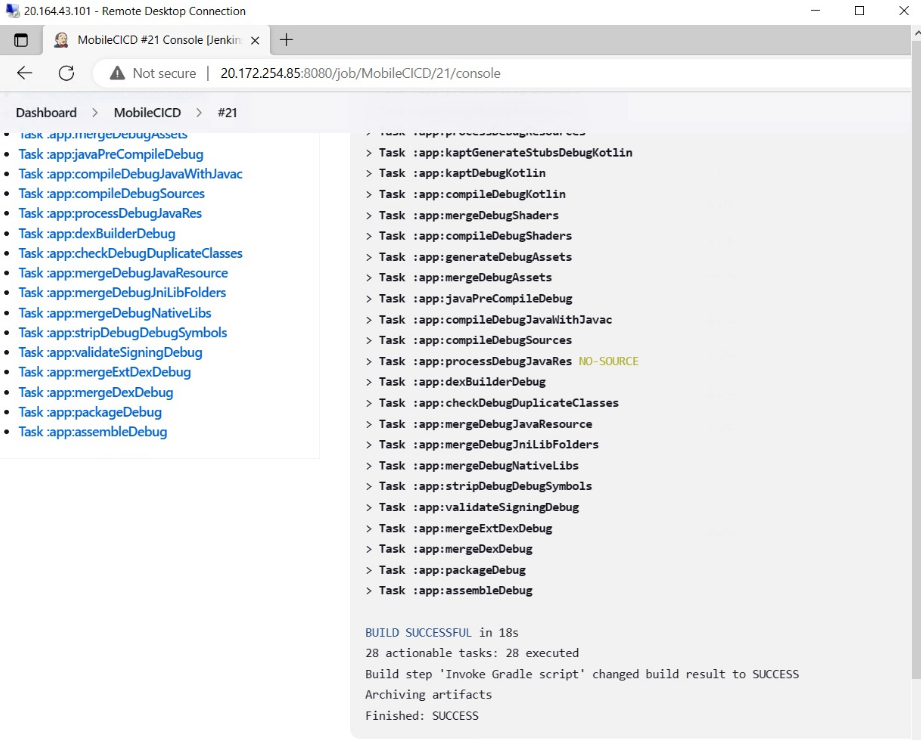
* Create a repository for sample code

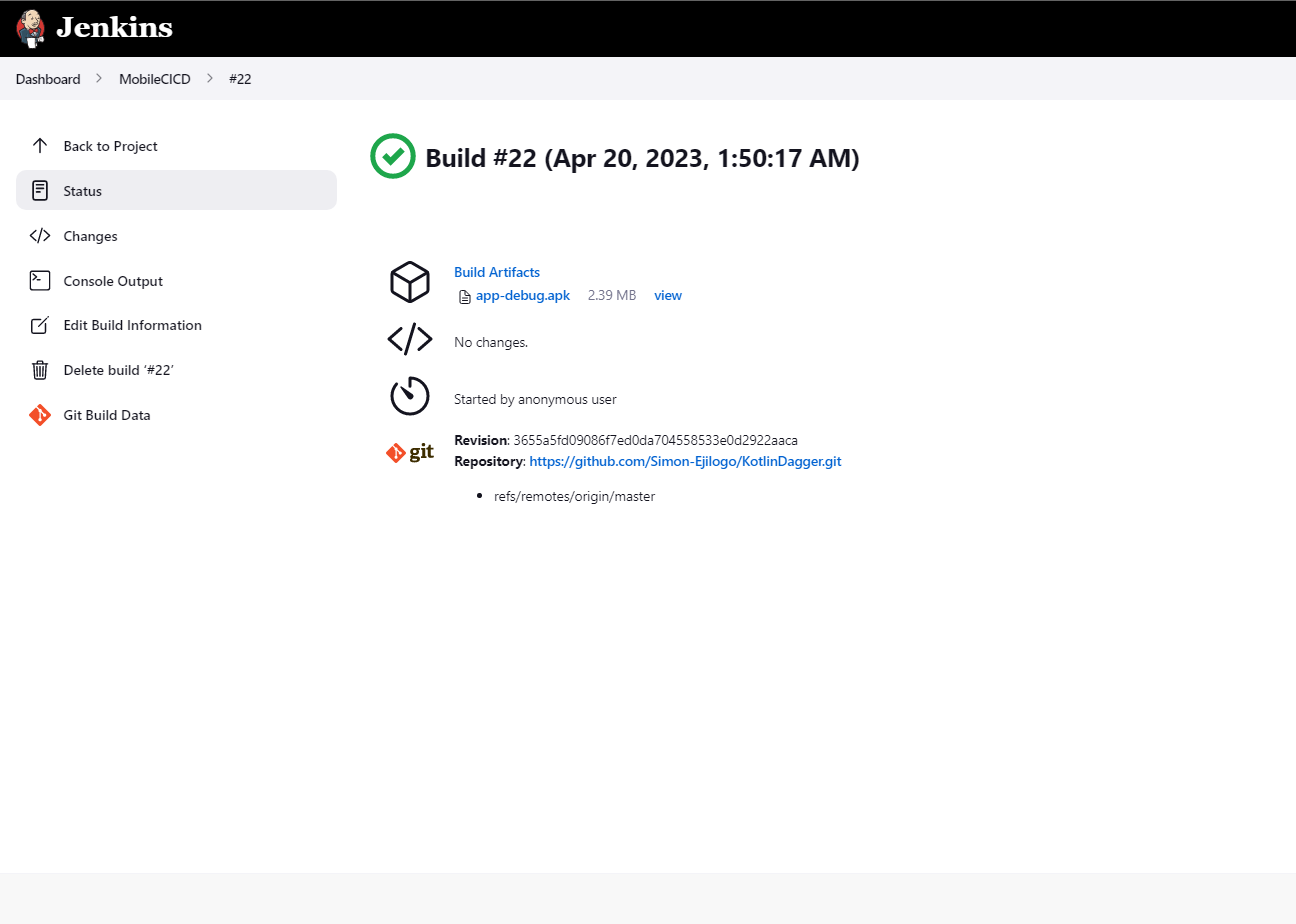
<https://github.com/Simon-Ejilogo/KotlinDagger.git>

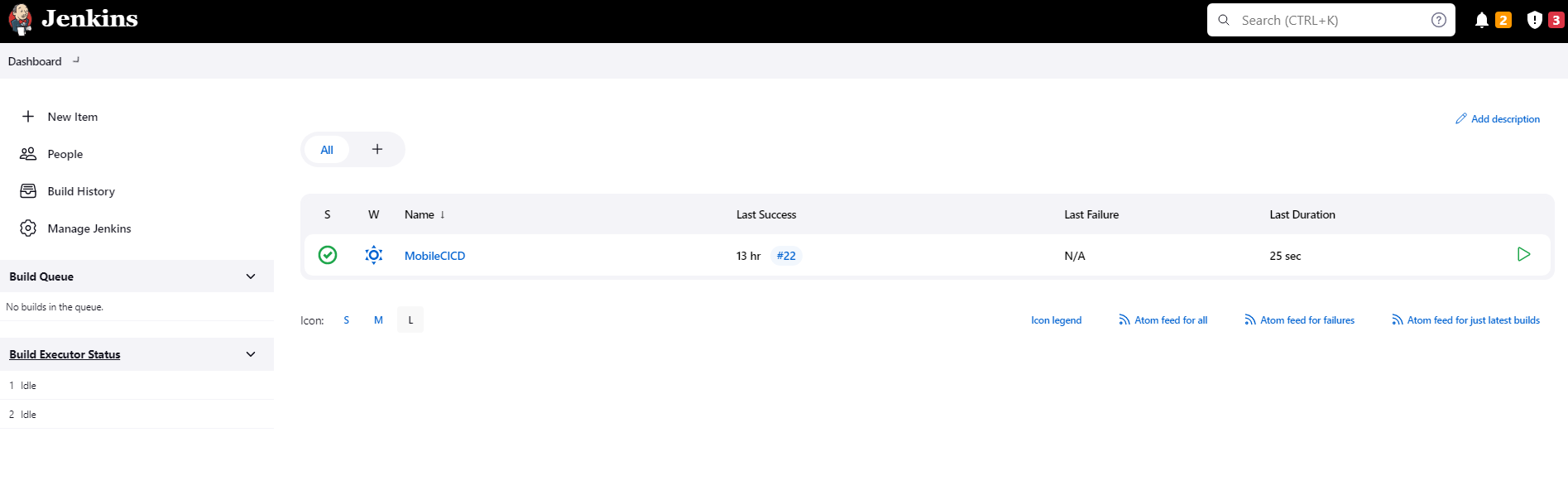
* Install the GitHub plugin in Jenkins: Go to "Manage Jenkins" > "Manage Plugins" > "Available" and search for "GitHub". Install the plugin and restart Jenkins.
* Create a personal access token in GitHub: Go to your GitHub account settings > Developer settings > Personal access tokens > Generate new token. Give the token a name and select the appropriate scopes.
* Configure GitHub credentials in Jenkins: Go to "Manage Jenkins" > "Manage Credentials" > "Jenkins" > "Global credentials" > "Add Credentials". Select "Username with password" as the kind and enter your GitHub username and access token as the username and password, respectively.
* Create a new Jenkins job: Click on "New Item" in the Jenkins dashboard and enter a name for your job. Select "Freestyle project" and click "OK".
* Configure the job: Under the "Source Code Management" section, select "Git" and enter the URL of your GitHub repository. Select the credentials you created earlier and specify the branch you want to build. Under "Build Triggers", select "GitHub hook trigger for GITScm polling".
* Configure the build: Under the "Build" section, add a build step to build your Android application. This may involve running Gradle commands, building an APK, or running automated tests. You can also configure post-build actions to deploy the app to a remote server or a mobile device.



* Save and run the job: Click "Save" to save the job configuration and then click "Build Now" to start the build process. Jenkins will check out the source code from GitHub, build the app, and report the results of the build. You can view the build log and any build artifacts by clicking on the job in the Jenkins dashboard.

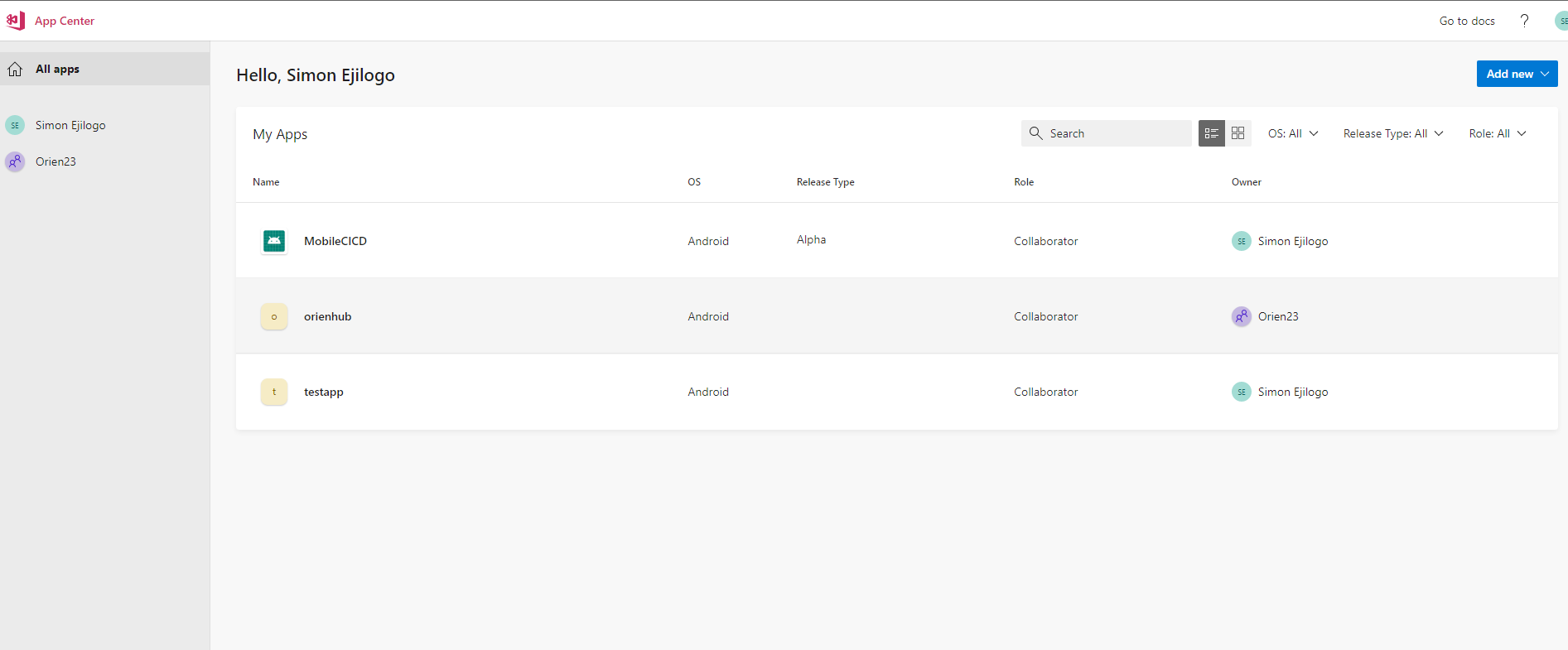


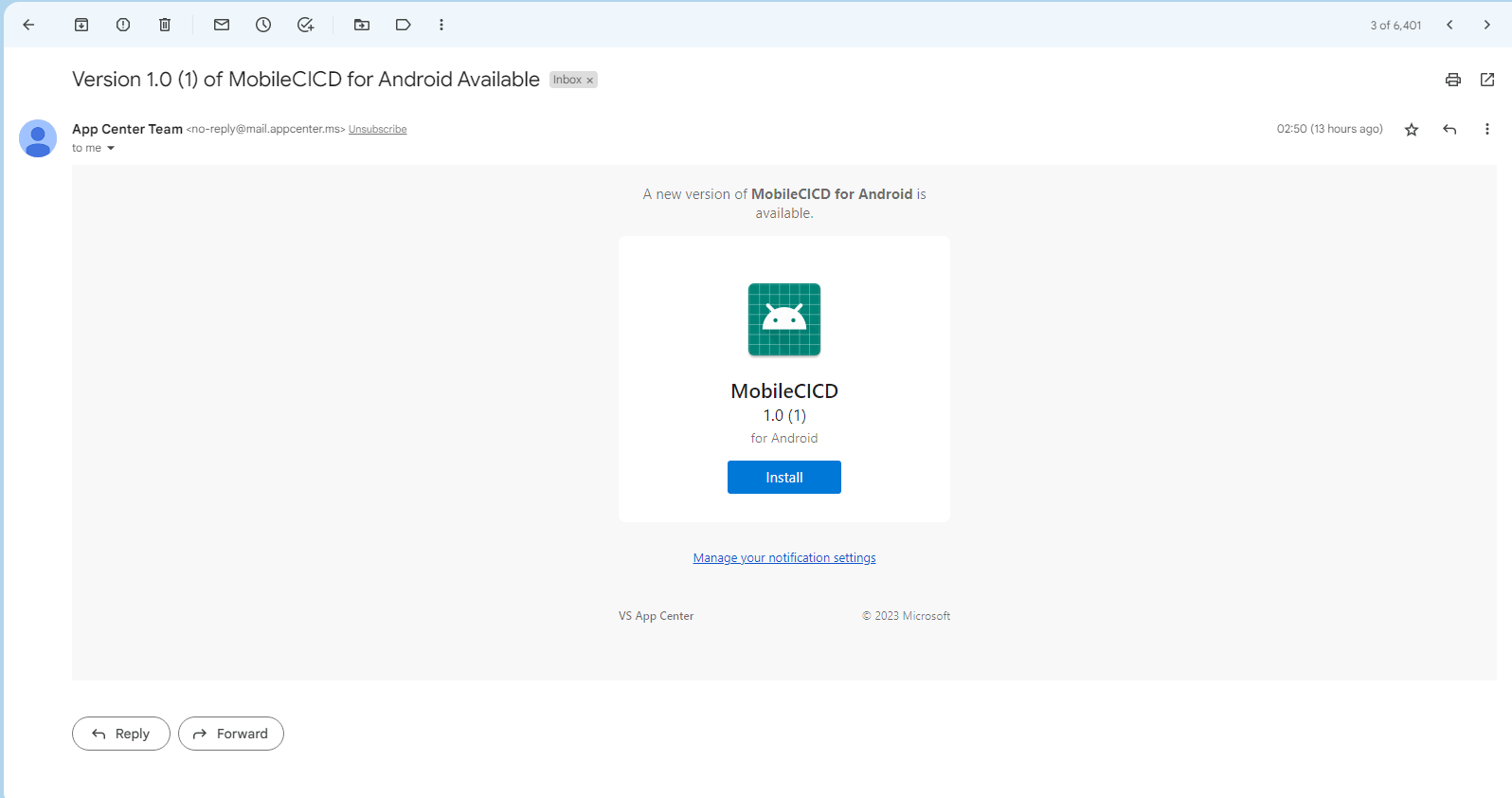




**CONNECT JENKINS TO APP CENTER AND DEPLOY BUILD .APK**

* Create an account on App Center: If you don't have an account already, create one by visiting the App Center website and signing up.
* Create an app in App Center: After logging in to App Center, click the "Add new app" button and follow the prompts to create a new app. Select "Android" as the platform and give your app a name.
* Generate an API token in App Center: Go to "Account settings" > "Api tokens" and click "New API token". Give the token a name and select the appropriate permissions for the token.
* Install the App Center plugin in Jenkins: Go to "Manage Jenkins" > "Manage Plugins" > "Available" and search for "App Center". Install the plugin and restart Jenkins.
* Configure App Center credentials in Jenkins: Go to "Manage Jenkins" > "Manage Credentials" > "Jenkins" > "Global credentials" > "Add Credentials". Select "Secret text" as the kind and enter your App Center API token as the secret. Give the credentials a meaningful ID.
* Configure the job: Under the "Post-build Actions" section, select "Upload to App Center". Enter your App Center organization name and app name, and select the distribution group(s) that you want to deploy to. Specify the APK file(s) that you want to deploy.
* Configure App Center API token in Jenkins: Under the "Post-build Actions" section, click the "Add" button next to "App Center API token". Enter the credentials ID you created earlier.
* Save and run the job: Click "Save" to save the job configuration and then click "Build Now" to start the build process. Jenkins will build your Android application and upload it to App Center for distribution to your selected groups.





* From the app center you can now share your app for testing by users.