

DevOps x AWS

Series III

Secure dependencies with CodeArtifact



Dahri Hadri

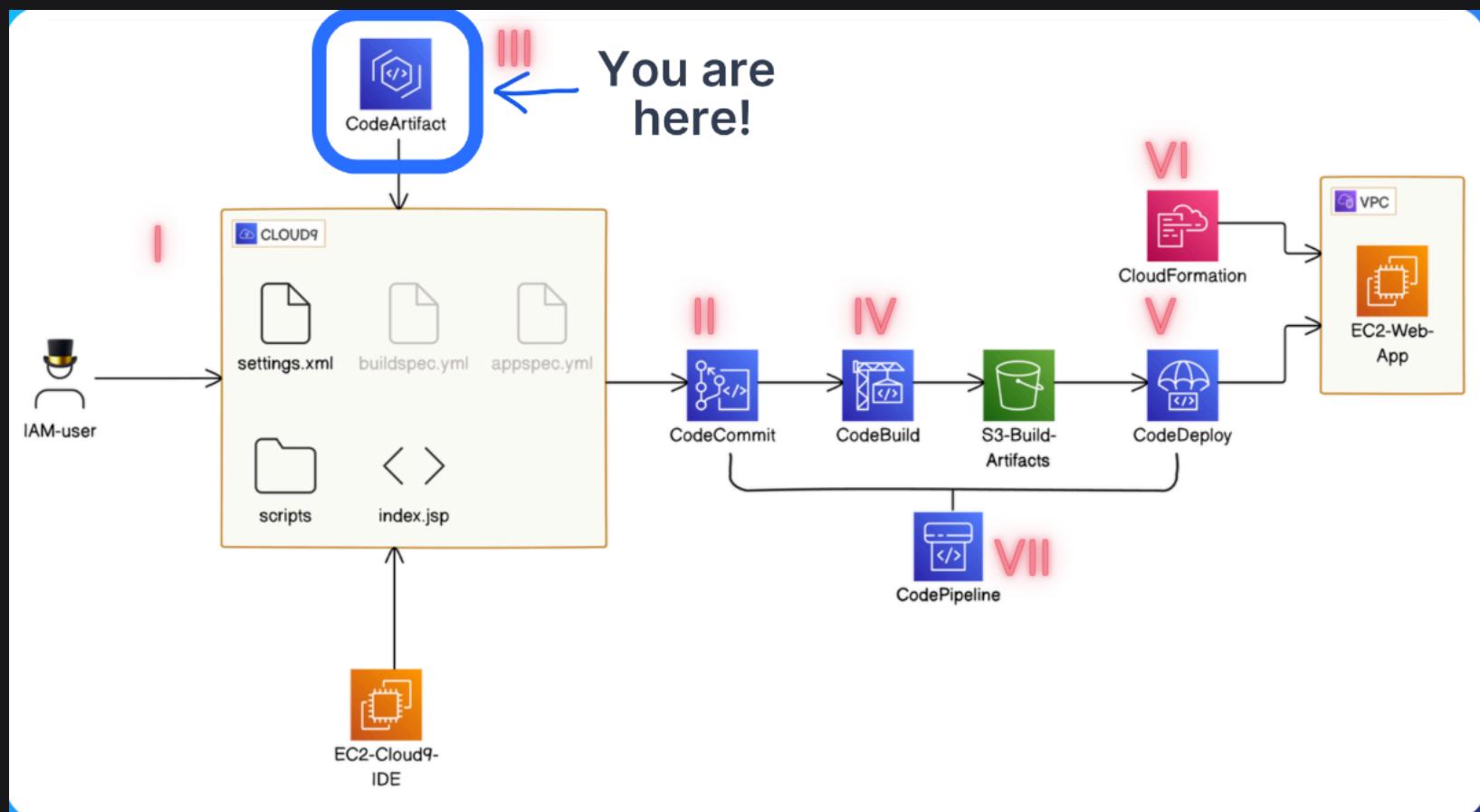


Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS

For this DevOps x AWS series, I am sharing 7 projects. In this SEVEN-project series, I will create a CI/CD pipeline to build and deploy a simple web application using AWS Code services. Here's what I'll build at the end of ALL seven projects:

- I. Set up a Web App + IDE with Cloud9 
- II. Set Up A Git Repository with AWS CodeCommit 
- III. Secure Project Dependencies with AWS CodeArtifact
- IV. Package an App with AWS CodeBuild
- V. Deploy an App with AWS CodeDeploy
- VI. Automate with AWS CloudFormation
- VII. CI/CD Pipeline with AWS CodePipeline





Dahri Hadri
linkedin.com/in/dahrihadri

Introducing AWS CodeArtifact!

What it does & how it's useful

AWS CodeArtifact is a secure, scalable package management service that centralizes dependency storage and management for software development.

Developers and teams use AWS CodeArtifact because it simplifies artifact management, enhances security through fine-grained access controls, and integrates seamlessly with existing CI/CD pipelines.

How I'm using it in today's project

I'm using AWS CodeArtifact in this project to securely manage and store software packages, ensuring consistent dependency versions across my development team.

This project took me...

This project took me approximately 40 minutes to complete.
Documentation took me around 40 minutes to write.



Dahri Hadri
linkedin.com/in/dahrihadri

Create a repository

- CodeArtifact is a fully managed artifact repository service that makes it easy to securely store, publish, and share software packages.
- The reason why I'm using CodeArtifact for my web app is to securely manage and can share software packages, streamline dependency management, and enhance collaboration.
- Instead of a single repository, there are actually three connected repositories that Maven uses to fetch packages.
 - First, there's my local repository, housing all software packages installed locally. Maven first checks here for required packages.
 - Second, the public upstream repository, known as "maven-central-store." Maven queries this when packages aren't found locally
 - Third, the Maven Central Repository stores extensive tools and resources. Maven accesses this repository if a needed package isn't in maven-central-store.

Package flow illustrating the connections between the three repositories.

Review and create Info

Package flow

Review how packages flow from external connections through nextwork to nextwork-packages.

External connections

public:maven-central

Domain: nextwork

Upstream repository ?
maven-central-store

Repository
nextwork-packages

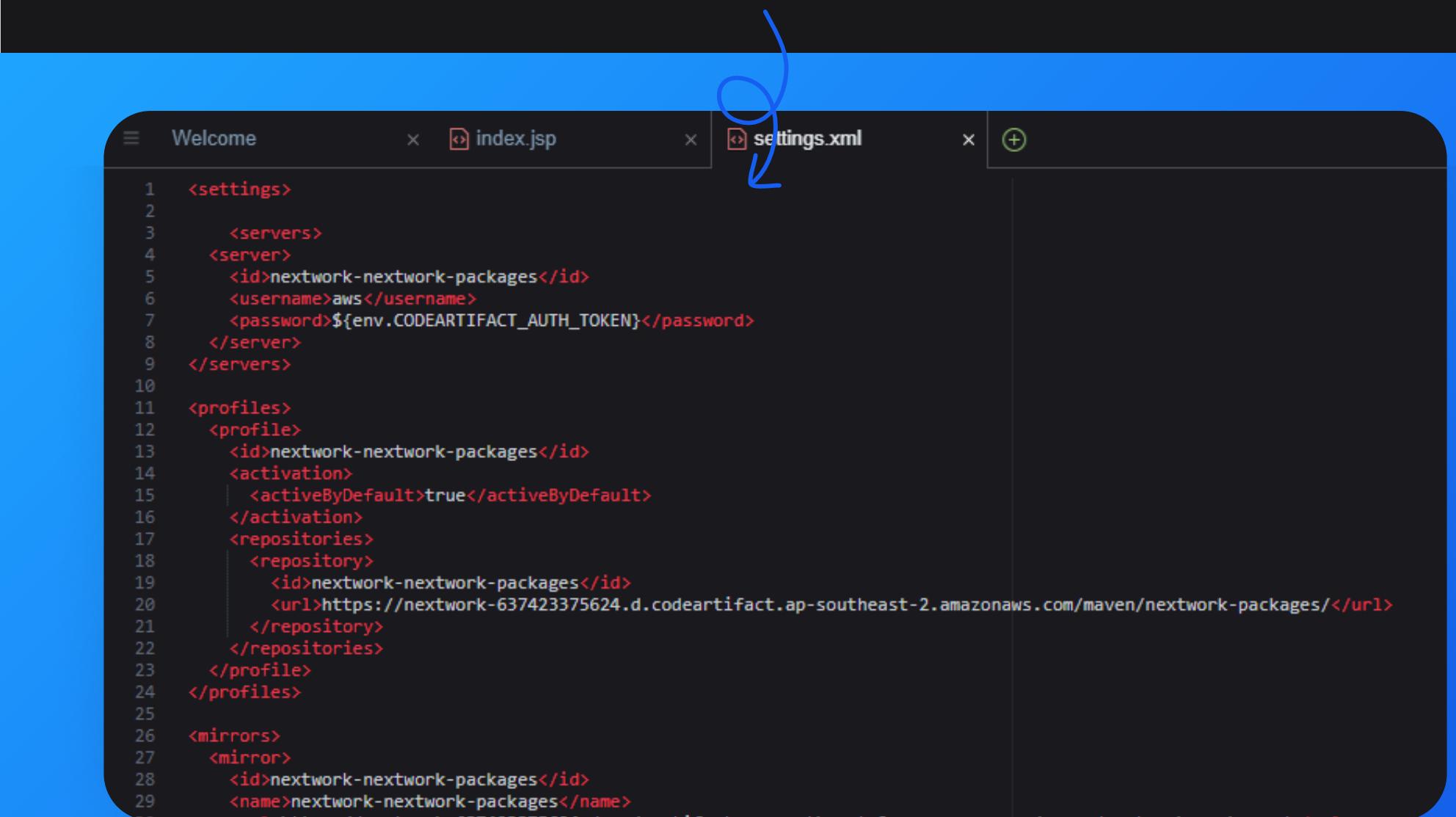


Dahri Hadri
linkedin.com/in/dahrihadri

Connecting my project to CodeArtifact

- Next, I connected my Cloud9 IDE to CodeArtifact so that I could seamlessly manage and access dependencies for my project, ensuring reliable and consistent builds across different development environments.
- I created a new file, `settings.xml`, in my web app. `settings.xml` is a Maven configuration file used to define settings such as repositories, plugin configurations, and authentication details, ensuring consistent and reliable build processes across development environments.
- The code I pasted into `settings.xml` were provided by CodeArtifact, so I did not have to write from scratch. The snippets of code stores authentication tokens to CodeArtifact and defines Maven will visit which repository,

My `settings.xml` file.



```
1 <settings>
2   <servers>
3     <server>
4       <id>nextwork-nextwork-packages</id>
5       <username>aws</username>
6       <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
7     </server>
8   </servers>
9
10  <profiles>
11    <profile>
12      <id>nextwork-nextwork-packages</id>
13      <activation>
14        <activeByDefault>true</activeByDefault>
15      </activation>
16      <repositories>
17        <repository>
18          <id>nextwork-nextwork-packages</id>
19          <url>https://nextwork-637423375624.d.codeartifact.ap-southeast-2.amazonaws.com/maven/nextwork-packages/</url>
20        </repository>
21      </repositories>
22    </profile>
23  </profiles>
24
25  <mirrors>
26    <mirror>
27      <id>nextwork-nextwork-packages</id>
28      <name>nextwork-nextwork-packages</name>
29      <url>https://nextwork-637423375624.d.codeartifact.ap-southeast-2.amazonaws.com/maven/nextwork-packages/</url>
30    </mirror>
31  </mirrors>
```

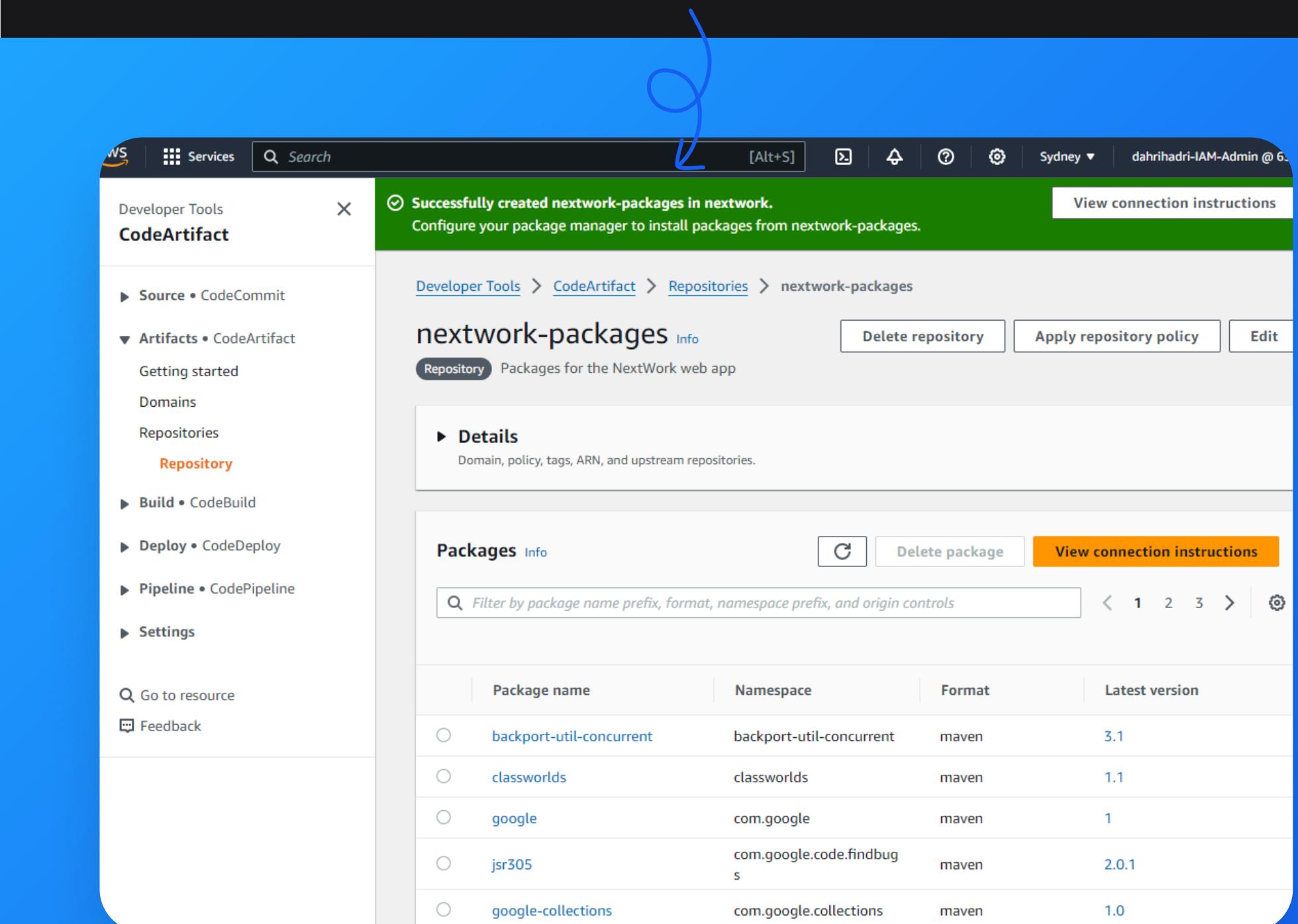


Dahri Hadri
linkedin.com/in/dahrihadri

Test the connection

- To test the connection between Cloud9 and CodeArtifact, I compiled my web app. Compiling means translating source code into executable code or intermediate code that can be run on a computer, ensuring all dependencies are resolved from CodeArtifact.
- After compiling, I checked my CodeArtifact repository. It contained the downloaded dependencies that been fetched.

My web app's packages popping up in my local repository.



The screenshot shows the AWS CodeArtifact console interface. A green success message at the top states: "Successfully created nextwork-packages in nextwork. Configure your package manager to install packages from nextwork-packages." Below this, the "nextwork-packages" repository details are shown, including its ARN and a list of packages. A blue arrow points from the text "My web app's packages popping up in my local repository." to the "Packages" table below.

Package name	Namespace	Format	Latest version
backport-util-concurrent	backport-util-concurrent	maven	3.1
classworlds	classworlds	maven	1.1
google	com.google	maven	1
jsr305	com.google.code.findbug.s	maven	2.0.1
google-collections	com.google.collections	maven	1.0

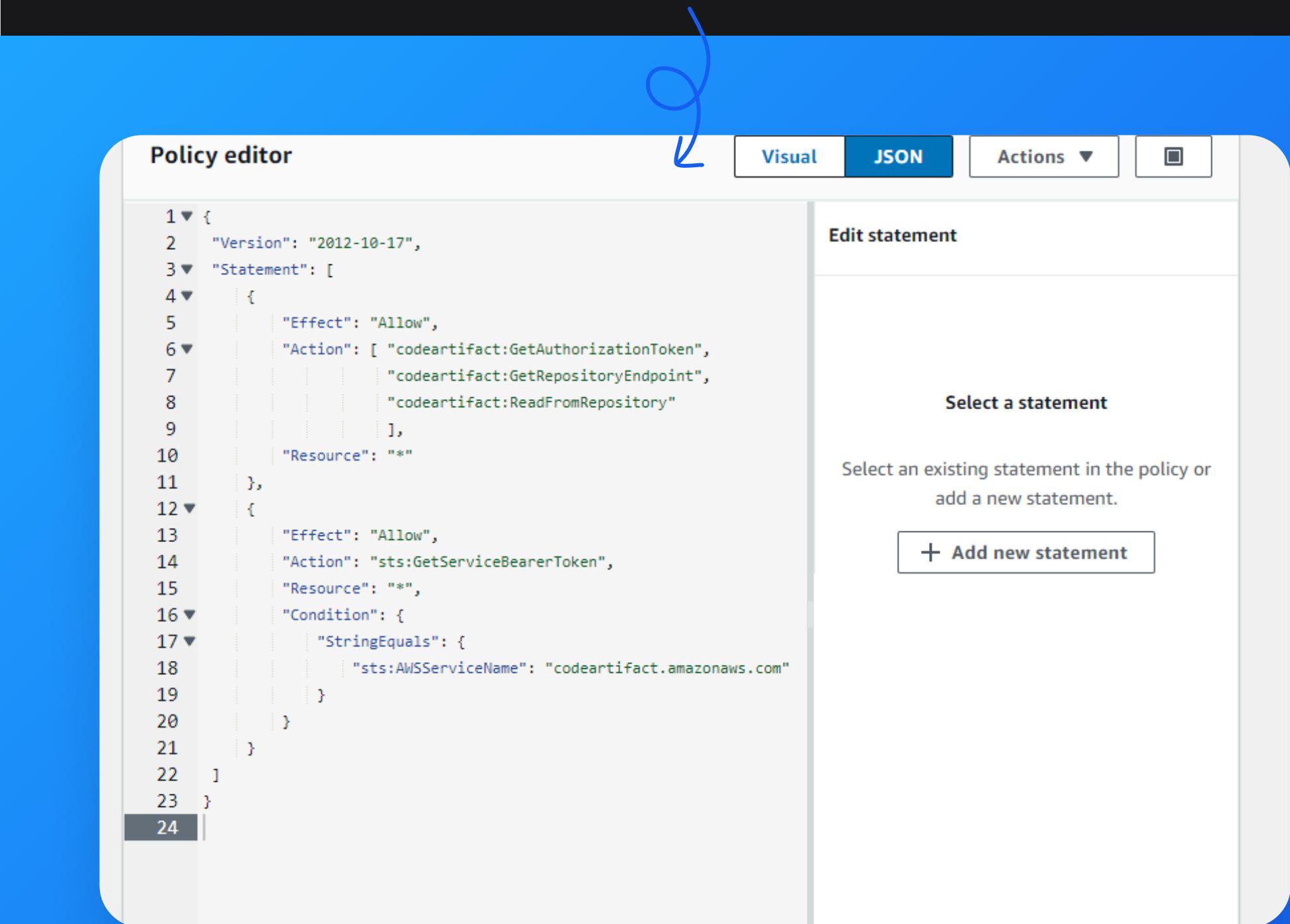


Dahri Hadri
linkedin.com/in/dahrihadri

Create IAM policies

- I also created an IAM policy because it defines specific permissions for accessing AWS CodeArtifact repositories. This ensures secure and controlled management of artifact repositories within my AWS environment.
- I defined my IAM policy using JSON. This policy allows:
 - `codeartifact:GetAuthorizationToken`, `codeartifact:GetRepositoryEndpoint`, and `codeartifact:ReadFromRepository` actions for accessing CodeArtifact resources.
 - `sts:GetServiceBearerToken` action for retrieving service bearer tokens, specifically for `codeartifact.amazonaws.com`.
- These permissions enable fetching authorization tokens, accessing repository endpoints, reading from repositories, and obtaining service bearer tokens for AWS CodeArtifact operations.

A peek at the policy that will provide access to CodeArtifact



The screenshot shows the AWS IAM Policy Editor interface. At the top, there are tabs for "Visual", "JSON" (which is selected), and "Actions". Below the tabs, the title "Policy editor" is visible. The main area contains a JSON code editor with a blue background and white text. The JSON code defines a policy with two statements. The first statement allows "codeartifact" actions on all repositories for all users. The second statement, under a condition, allows "sts:GetServiceBearerToken" for the "codeartifact.amazonaws.com" service. A blue arrow points from the text "A peek at the policy that will provide access to CodeArtifact" to the "Edit statement" section of the policy editor. The "Edit statement" section includes a "Select a statement" button, a note to "Select an existing statement in the policy or add a new statement", and a "Add new statement" button.

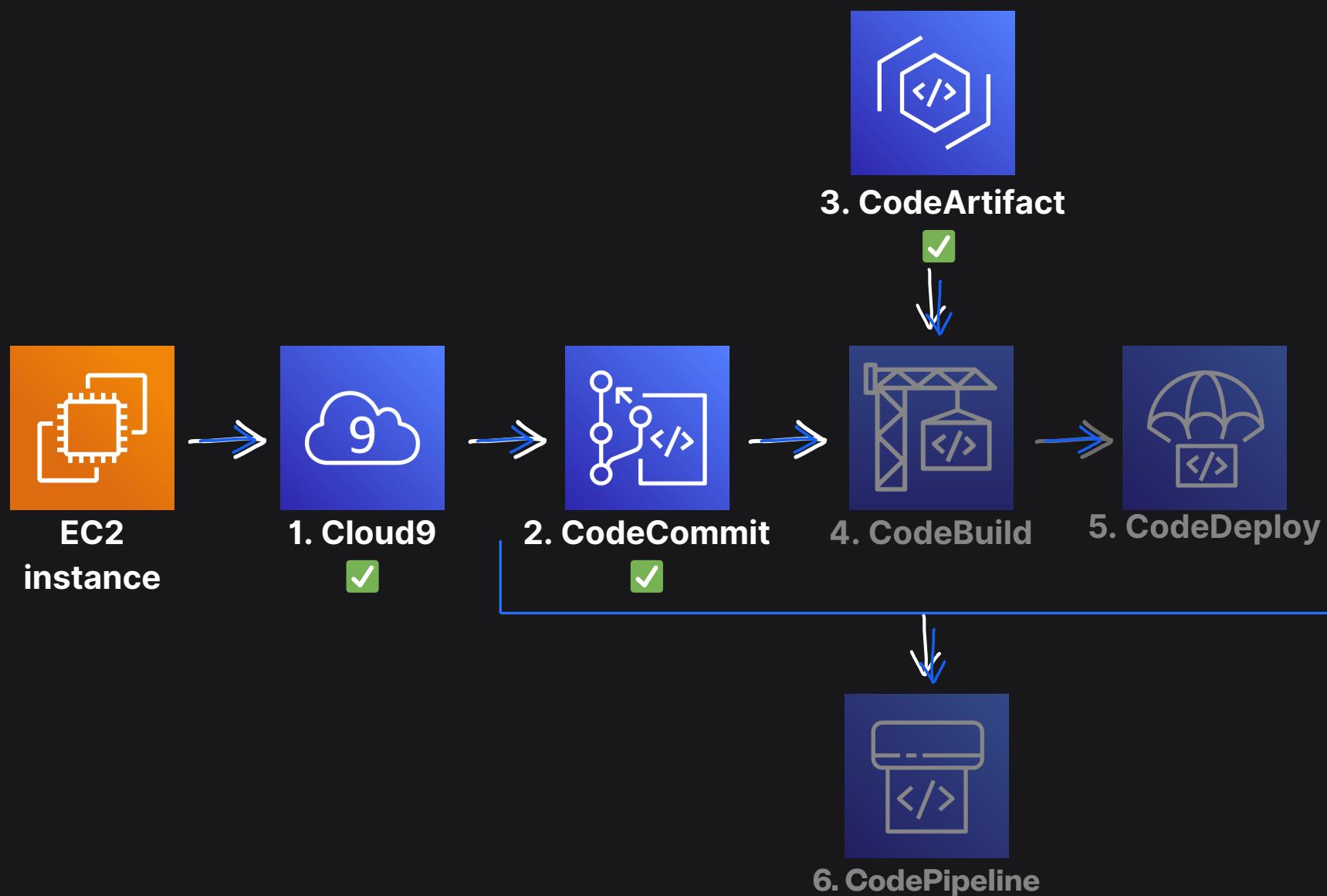
```
1 ▼ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [ "codeartifact:GetAuthorizationToken",  
7                         "codeartifact:GetRepositoryEndpoint",  
8                         "codeartifact:ReadFromRepository"  
9                     ],  
10            "Resource": "*"  
11        },  
12        {  
13            "Effect": "Allow",  
14            "Action": "sts:GetServiceBearerToken",  
15            "Resource": "*",  
16            "Condition": {  
17                "StringEquals": {  
18                    "sts:AWSServiceName": "codeartifact.amazonaws.com"  
19                }  
20            }  
21        }  
22    ]  
23}  
24
```



Dahri Hadri
linkedin.com/in/dahrihadri

My CI/CD pipeline so far...

1. **AWS Cloud9** is responsible for IDE setup and development environment management.
2. **AWS CodeCommit** is responsible for hosting secure and scalable Git repositories in the AWS cloud, facilitating collaborative software development workflows.
3. **AWS CodeArtifact** is responsible for securely storing and managing software packages and dependencies, ensuring reliable and scalable artifact management.





Dahri Hadri
linkedin.com/in/dahrihadri

My key learnings

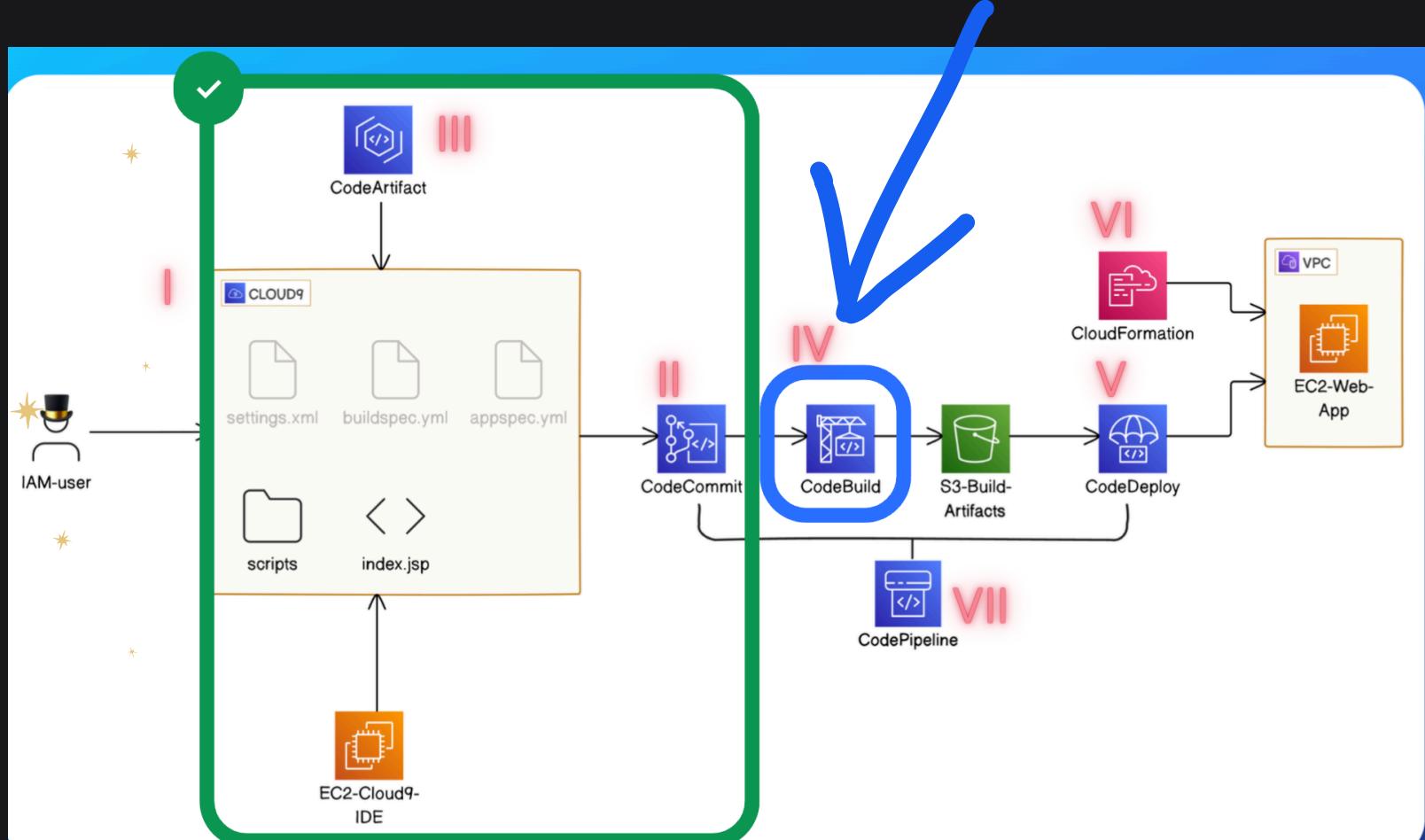
- 1 A public upstream repository is a centralized source for accessing shared software packages and dependencies from external sources.
- 2 `settings.xml` is a configuration file used to define Maven settings, including repositories like AWS CodeArtifact, for fetching project dependencies efficiently.
- 3 To test the connection between Cloud9 and CodeArtifact, I compiled my web app to verify dependency fetching and integration success.
- 4 One thing I didn't expect was the seamless integration and dependency management provided by AWS CodeArtifact, streamlining my development workflow significantly.



Dahri Hadri
linkedin.com/in/dahrihadri

Great! we are done with series III

coming up next in series
IV - CodeBuild!





NEXTWORK

Find this helpful?

-  Like this post
-  Leave a comment
-  Save for later
-  Let's connect!



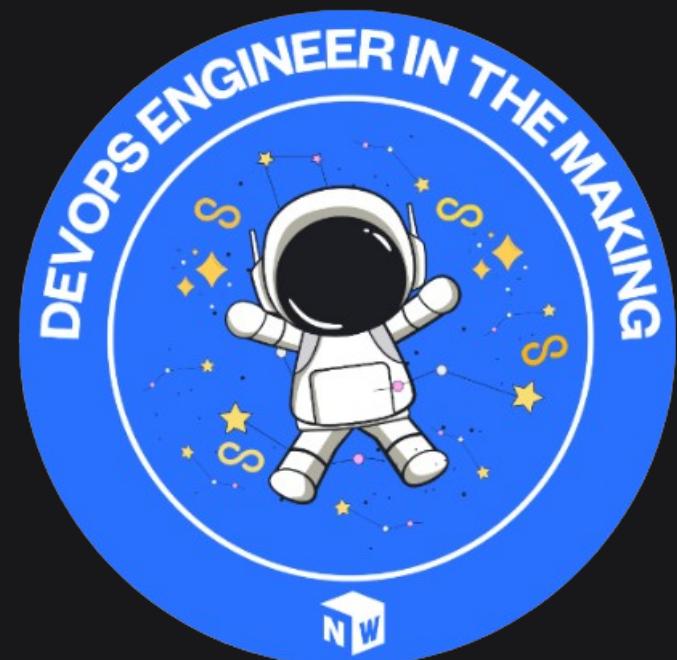
Dahri Hadri



@dahrihadri



<https://www.linkedin.com/in/dahrihadri>



Ask me about it

