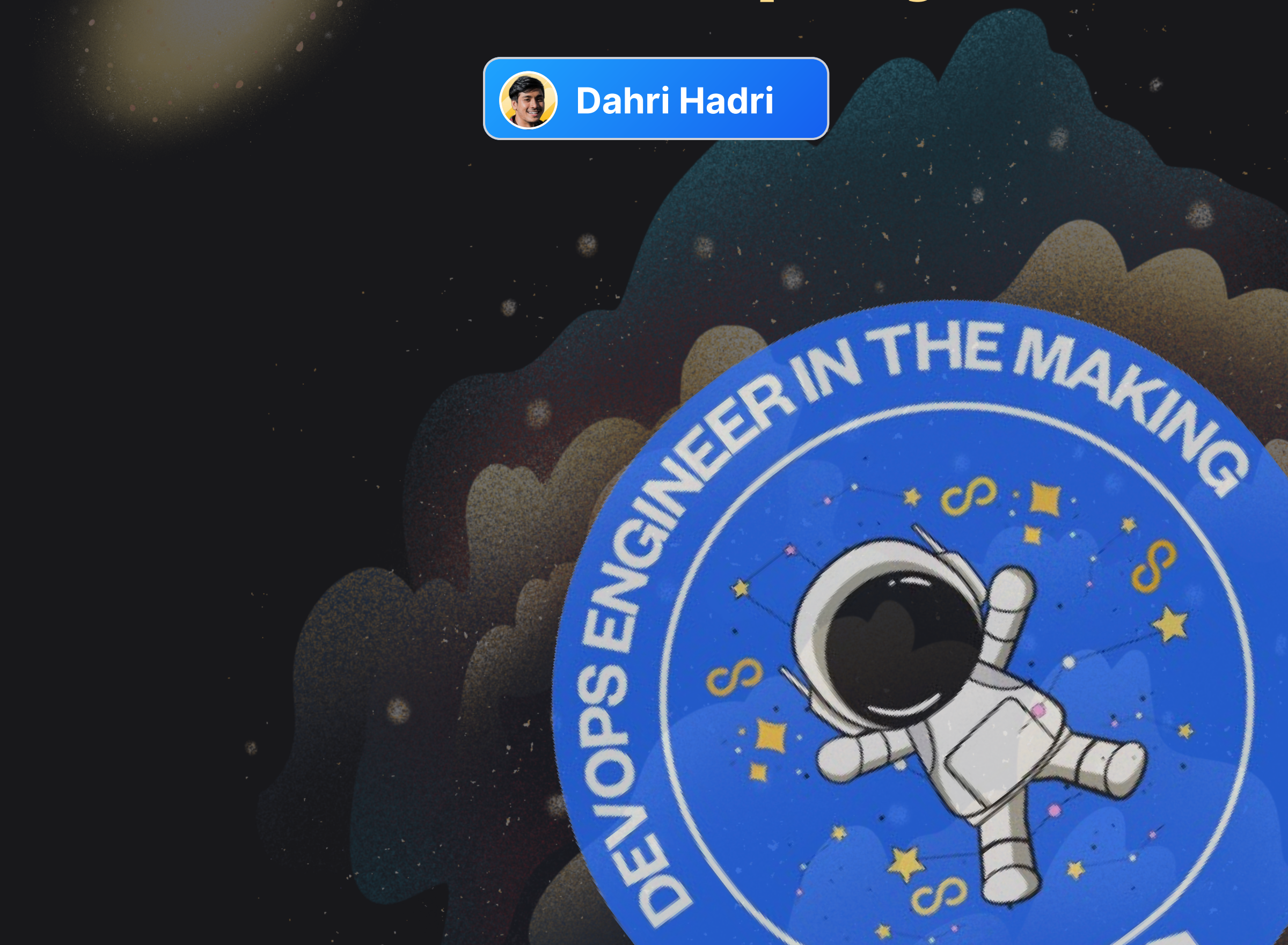
NEXTWORK

DevOps x AWS

Series V

Deploy an App with CodeDeploy

Dahri Hadri



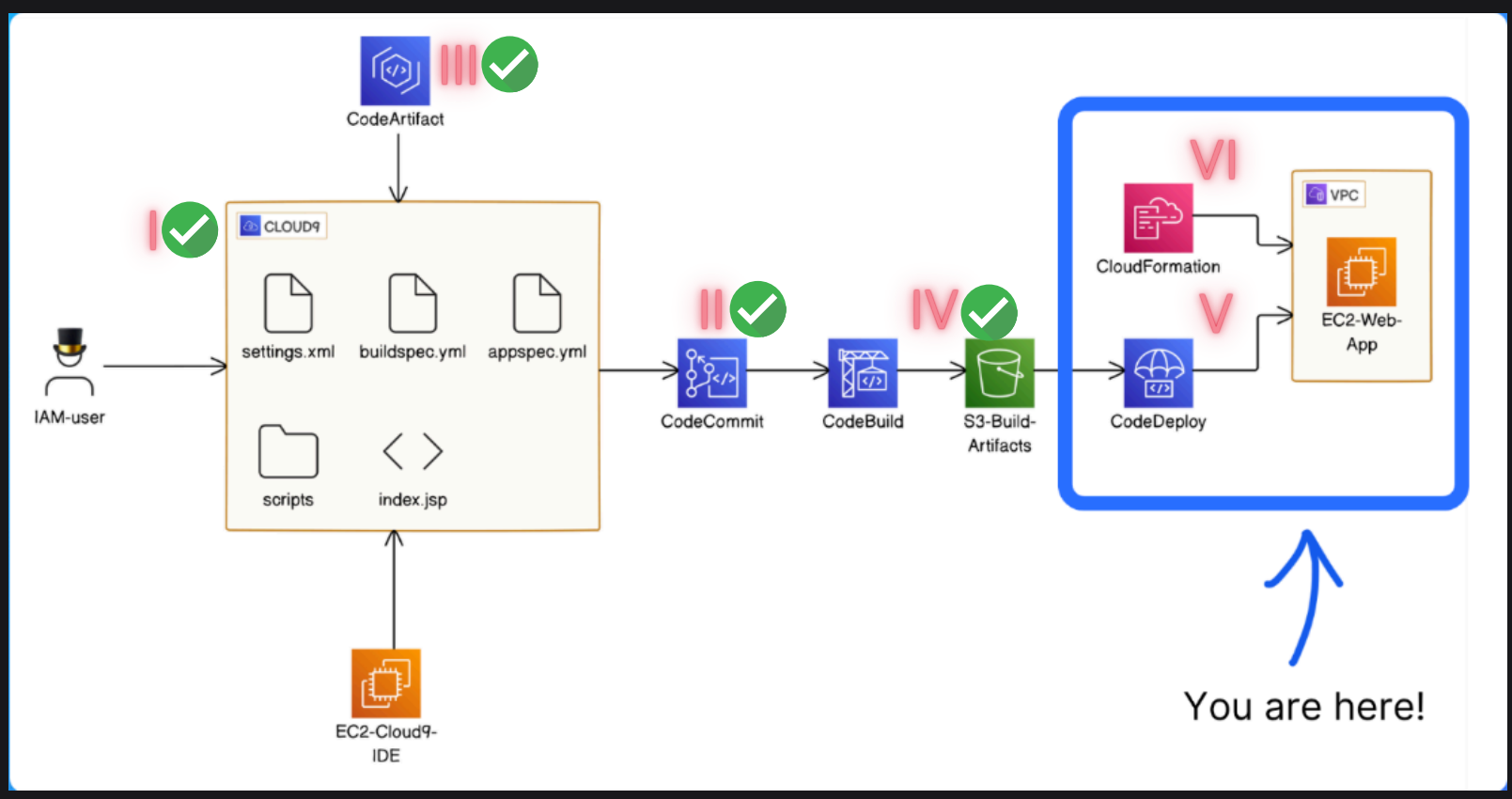
Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

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:

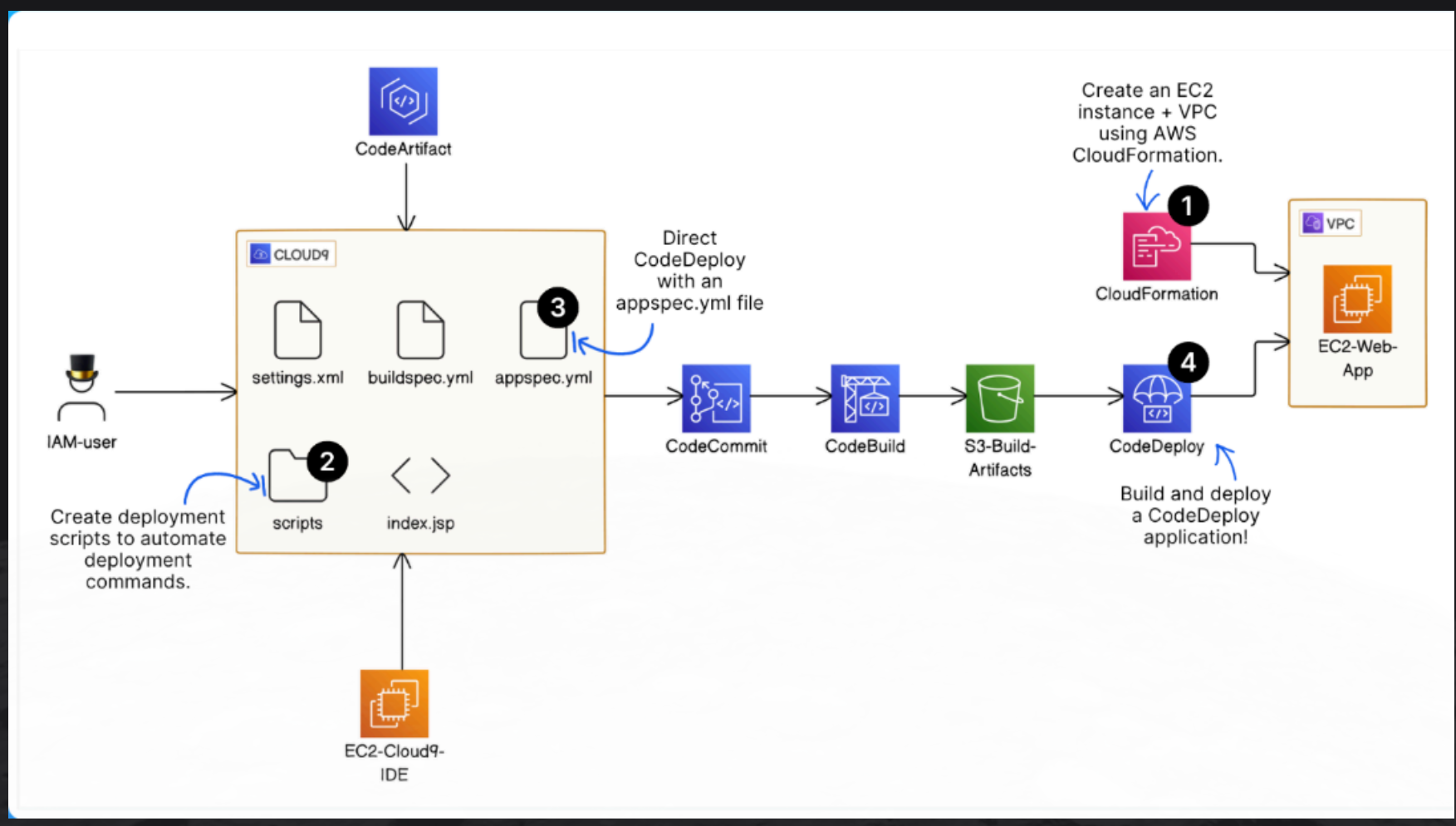
I. Set up a Web App + IDE with Cloud9
II. Set Up A Git Repository with AWS CodeCommit
III. Secure Project Dependencies with AWS CodeArtifact
IV. Package an App with AWS CodeBuild
V. Deploy an App witth AWS CodeDeploy
VI. Automate with AWS CloudFormation
VII. CI/CD Pipeline with AWS CodePipeline



Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

Our game plan for this project 🚀



Let's get ready to:

1. Create an EC2 instance and VPC with AWS CloudFormation.
2. Create deployment scripts in your project to automate deployment commands.
3. Direct CodeDeploy with a new appspec.yml file.
4. Build and deploy your CodeDeploy application!

Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

Introducing AWS CodeDeploy!

What it does & how it’s useful

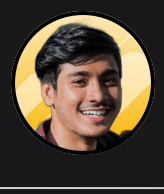
AWS CodeDeploy is a deployment service that automates code deployments to any instance, including EC2 instances and on-premises servers.
Developers and teams use AWS CodeDeploy because it simplifies deploying and updating applications, ensuring consistent, reliable, and repeatable deployments across environments.

How I’m using it in today’s project

I’m using AWS CodeDeploy in this project to automate the deployment of a web application to my EC2 instance, ensuring a seamless and efficient update process.

This project took me...

This project took me about 1 hour to complete, including setting up AWS CodeBuild. Documentation took me around 40 minutes to write and polish, ensuring it’s clear and comprehensive for future reference.



Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

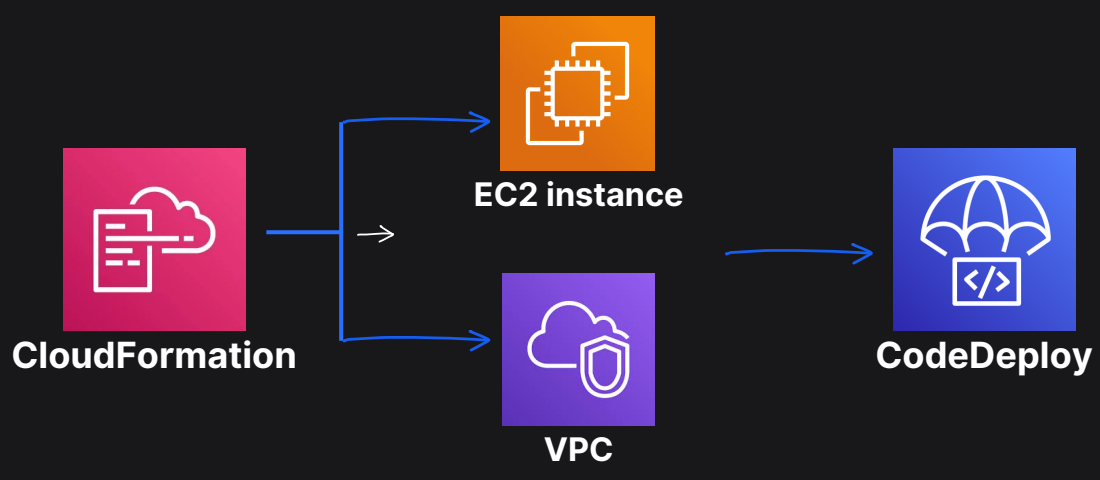
Set up an EC2 instance

- I set up an EC2 instance and VPC because to create a secure and scalable environment for deploying and running our web application, ensuring isolation and control over the network resources.

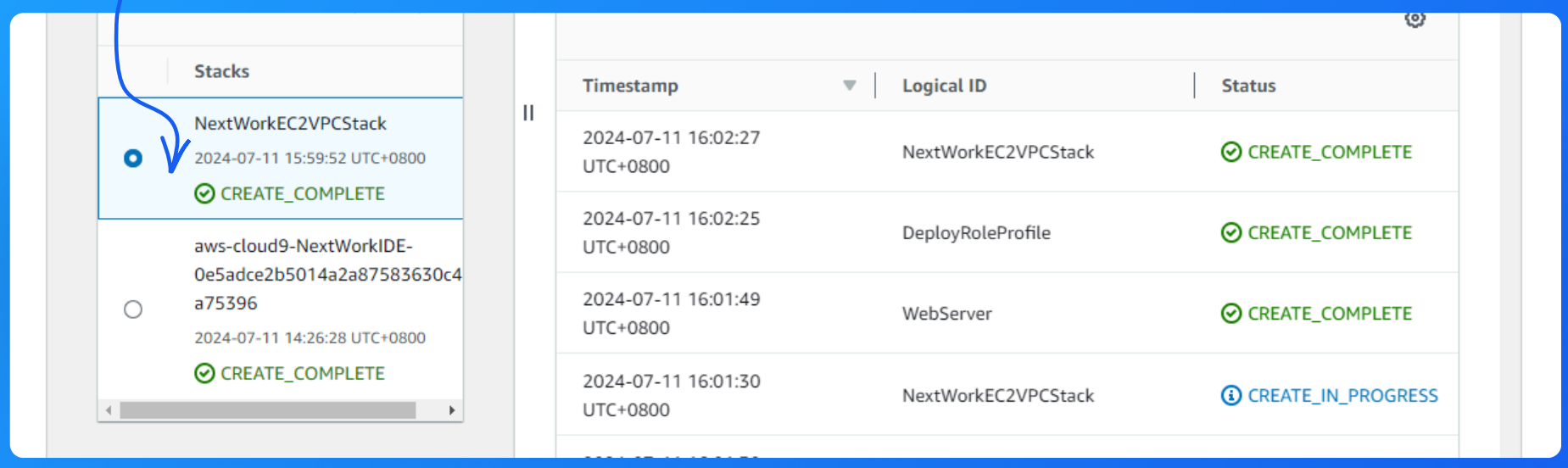
- The production environment is different to the build environment because the production environment hosts the live application accessible to end users, requiring stringent security, scalability, and reliability. In contrast, the build environment is for developing and testing the application before it's deployed to production.

- To set up my EC2 instance and VPC, I used AWS CloudFormation to automatically deploy my EC2 instance and VPC, streamlining the process and ensuring consistency across environments.

- The diagram below illustrates the architecture of my EC2 instance and VPC setup, highlighting the key components and their interactions within the AWS infrastructure.



A peek into my CloudFormation stack's deployment!



Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

Write bash scripts

- Scripts are automated instructions that perform specific tasks, making processes efficient and repeatable.

- Bash is a Unix shell and command language used for executing commands and writing scripts in Linux environments.

- I created 3 scripts for this the deployment process:
  a. Install_dependencies.sh: Installs Tomcat and Apache (httpd), and configures Apache as a reverse proxy for Tomcat.
  b. start_server.sh: Starts and enables Tomcat and Apache (httpd) services to run your web application.
  c. stop_server.sh: Stops Apache (httpd) and Tomcat services if they are running.

A peek into appspec.yml!

Welcome × index.js × settings × buildspe × install_(× start_se × stop_se × appspe ×

```yaml
version: 0.0
os: linux
files:
  - source: /target/nextwork-web-project.war
    destination: /usr/share/tomcat/webapps/
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

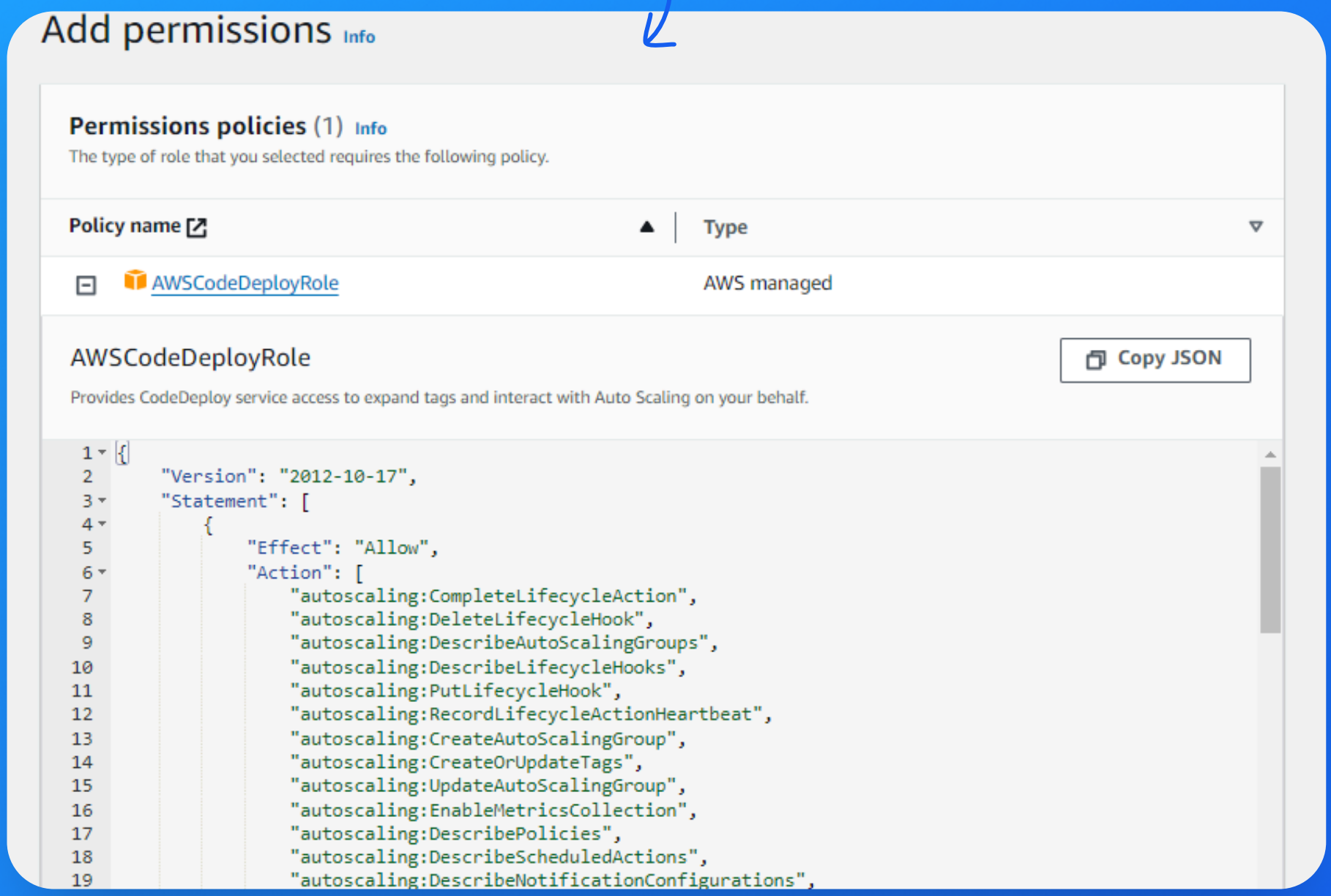
Dahri Hadri
linkedin.com/in/dahrihadri

CodeDeploy’s IAM Role

- I created an IAM service role for CodeDeploy because it provides the necessary permissions and access controls for CodeDeploy to manage deployments securely and effectively.

- To set up CodeDeploy's IAM role, I used AWS Management Console to attached a policy called "AWSCodeDeployRole". This policy automatically adds default permissions that CodeDeploy often needs.

The permissions granted by my CodeDeploy IAM role.



Dahri Hadri
linkedin.com/in/dahrihadri

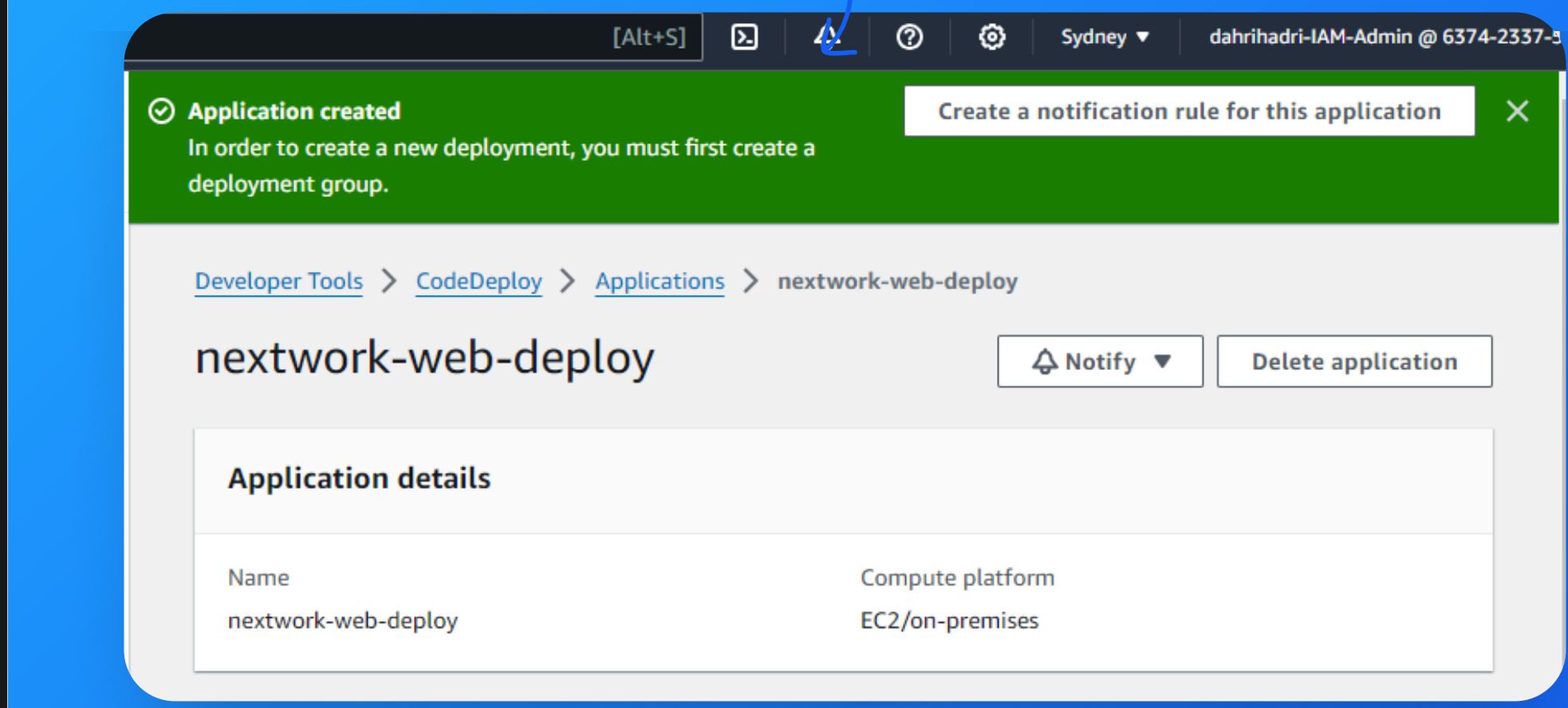
Create a CodeDeploy app

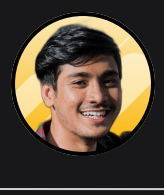
- A CodeDeploy application means a CodeDeploy application defines the resources and deployment settings for deploying a specific application to your instances.

- To create a CodeDeploy application, I had to select a compute platform, which means selecting a compute platform determines where your application will be deployed, whether it's on EC2 instances, Lambda functions, or an on-premises server.

- The compute platform I chose was EC2 instances, because I chose EC2 instances because my application is a web server that needs to run on virtual servers managed by AWS, providing flexibility and scalability.

My CodeDeploy app ready for a deployment!




Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

Set up a deployment group

- A deployment group means a set of instances targeted for a deployment. It defines where and how an application revision should be deployed.

- To create my deployment group, I set up a:
  - **Service role,** which is an IAM role granting permissions for CodeDeploy to interact with other AWS services.

  - **Deployment type,** which is how deployment will be managed, I chose in place, i.e. my webserver EC2 instance will deploy the latest web app right away without needing to create a new environment.

  - **Environment configuration**, which means specifies the type of instances and their settings where the application will be deployed.

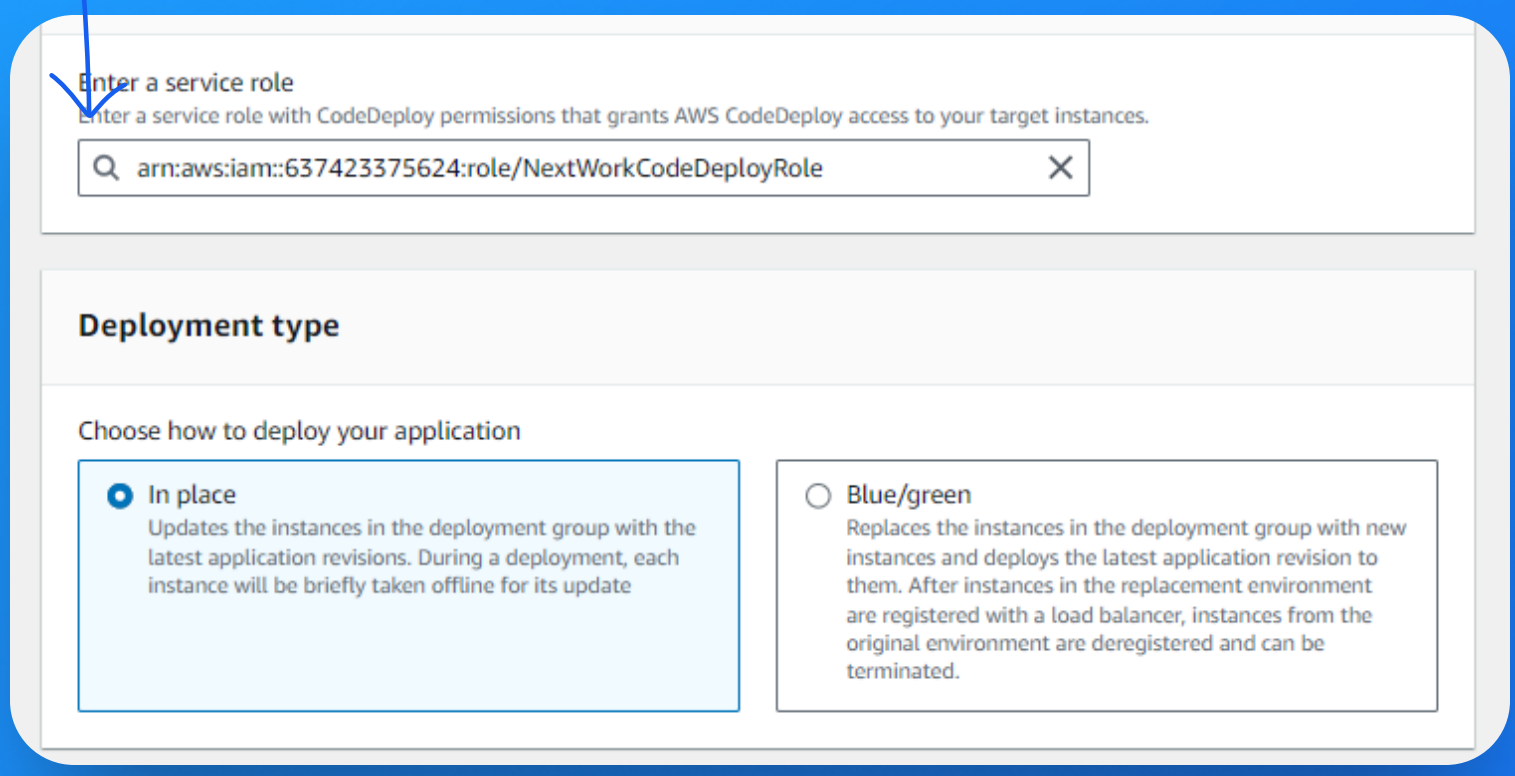
  - **CodeDeploy Agent,** which is a software installed on each instance to manage deployments and report back to CodeDeploy.

  - **Deployment settings**, which are specify how the deployment process is managed, such as all-at-once or rolling updates.

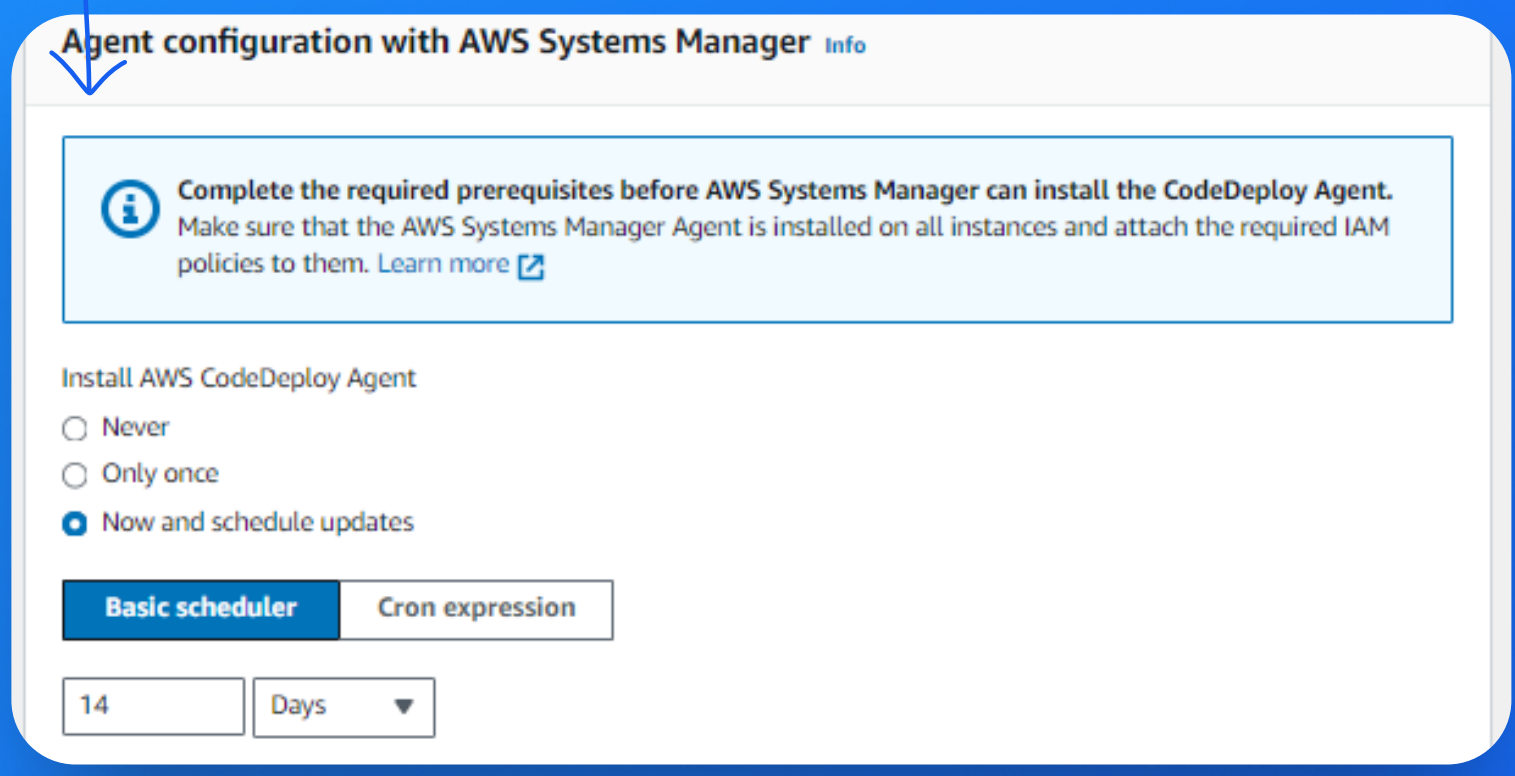
  - For the load balancer setting, I chose to uncheck Enable load balancing. This was because my application is currently deployed on a single EC2 instance without the need for load balancing.

Dahri Hadri
linkedin.com/in/dahrihadri

Setting up the service role + deployment type.



Setting up the CodeDeploy Agent.



Dahri Hadri
linkedin.com/in/dahrihadri

Deployment success! 🚀

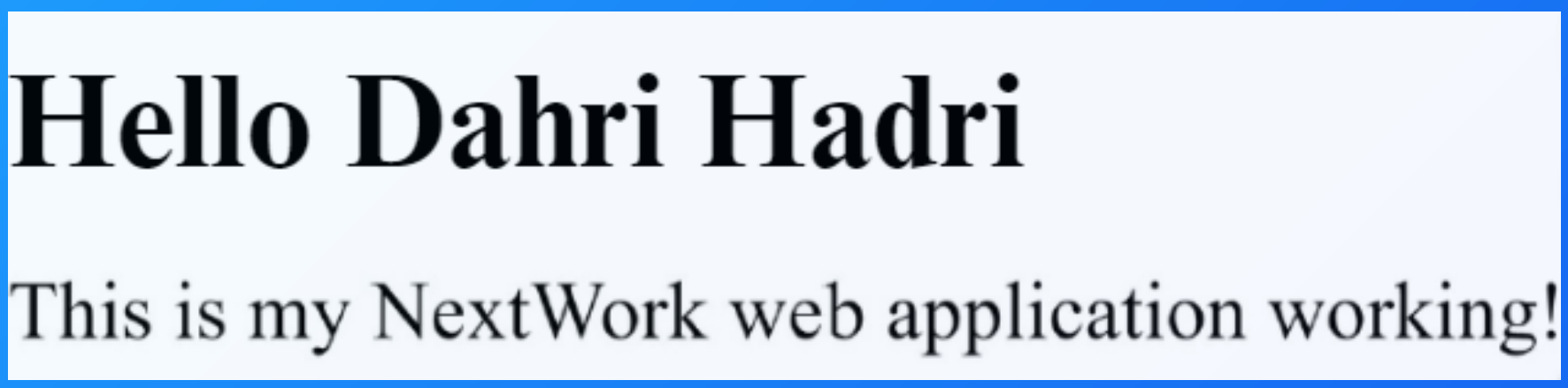
- After setting up my deployment group, I was ready to kick off my first deployment!

- To create my deployment, I had to set up a revision location, which means setting up a revision location involves specifying where CodeDeploy can find the application's deployment artifacts, typically stored in an S3 bucket.

- My revision location was my S3 URI pointing to my deployment artifact ZIP file.

- To visit my web app, I had to visit my EC2 instance's public IP address.

Amazing! I could see my web app taking shape.



Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

My CI/CD pipeline so far...

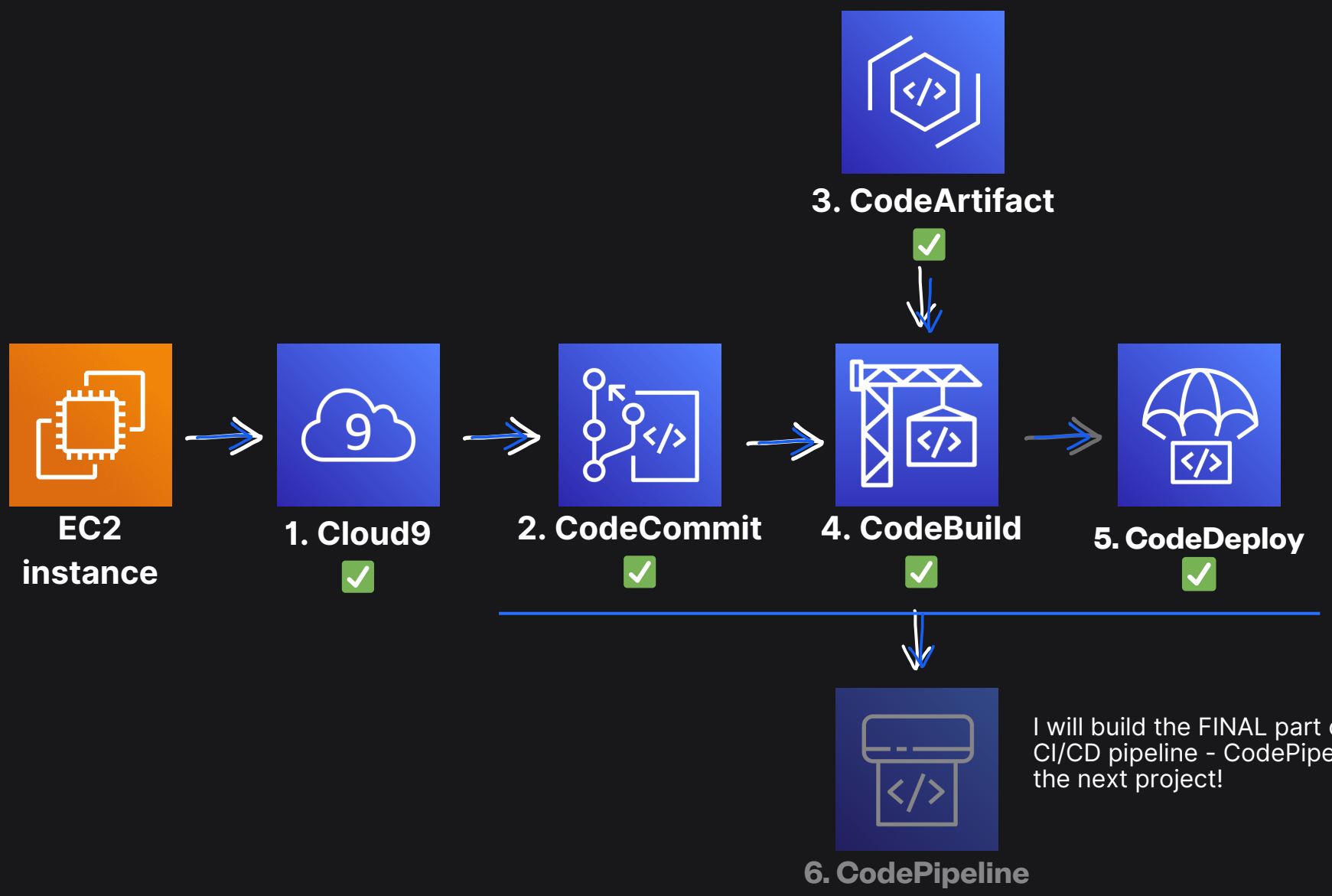
1. **AWS Cloud9** is responsible for IDE setup and development environment management.

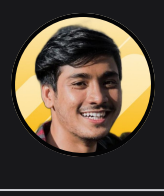
2. **AWS CodeCommit** is responsible for hosting secure and scalable Git repositories in the AWS cloud, facilitating collaborative software development workflows.

3. **AWS CodeArtifact** is responsible for securely storing and managing software packages and dependencies, ensuring reliable and scalable artifact management.

4. **AWS CodeBuild** is responsible for automating the build and testing of code in the cloud, providing scalable and efficient infrastructure for continuous integration and delivery (CI/CD) pipelines.

5. **AWS CodeDeploy** is responsible to automates deployment processes, ensuring consistent and reliable application updates across EC2 instances and on-premises servers.





Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

My key learnings

1. The deployment process means the method of building an application which includes, compiling, configuring and deploying.

2. I created a separate environment for deployment because isolating production from development ensures stability and reliability.

3. To create a deployment, I had to set up different resources in AWS, which included: an EC2 instance, a VPC, IAM roles, CodeDeploy application and deployment group, scripts for server management, and a CI/CD pipeline.

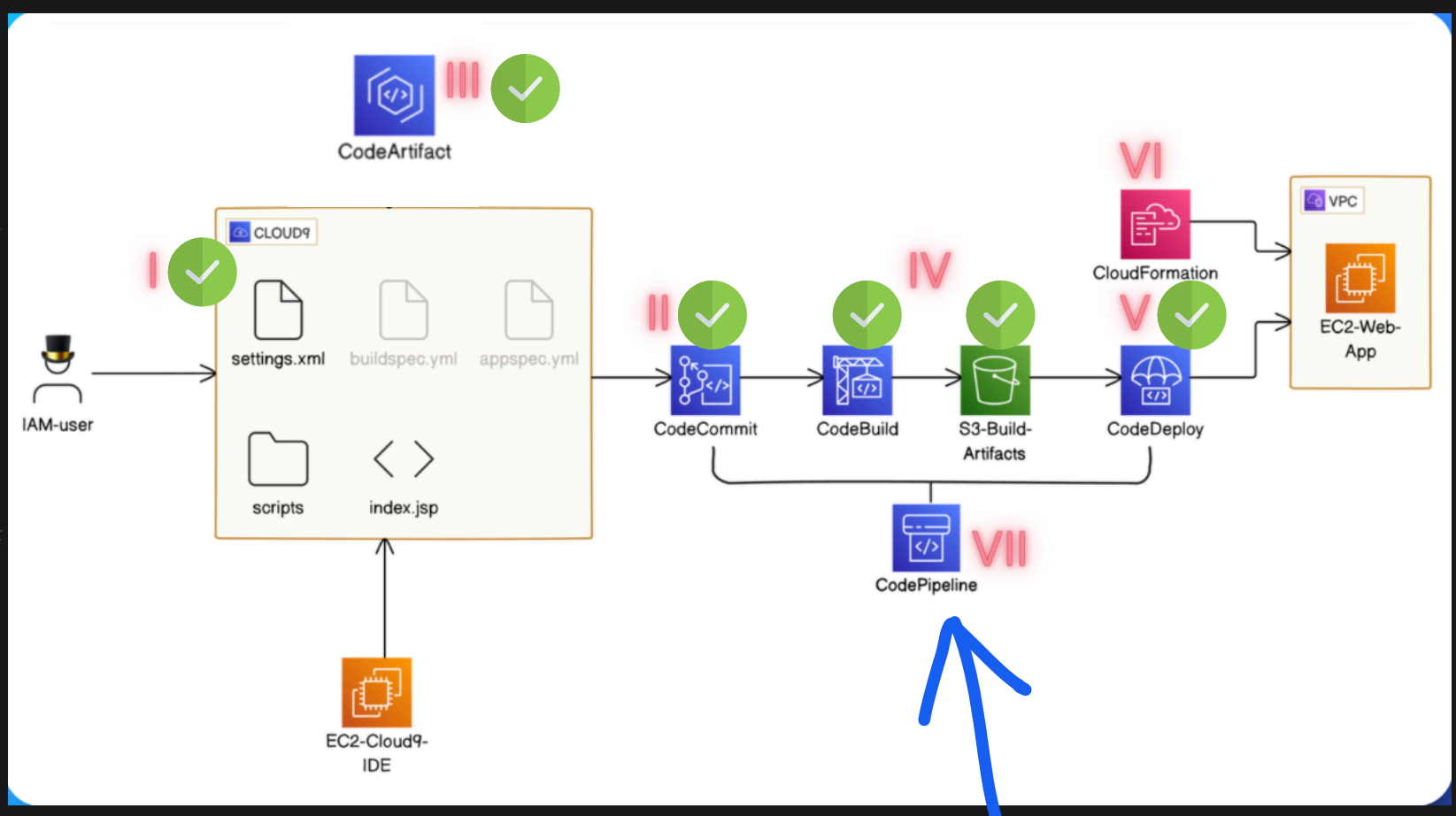
4. One thing I didn’t expect was the level of detail and configuration needed to manage server environments effectively during deployments.

Dahri Hadri
linkedin.com/in/dahrihadri

DevOps x AWS
Series V
NEXTWORK

Great!

we are done with series V



I will build the FINAL part of this CI/CD pipeline - CodePipeline - in the next project!

NEXTWORK

Find this helpful?

Like this post

Leave a comment

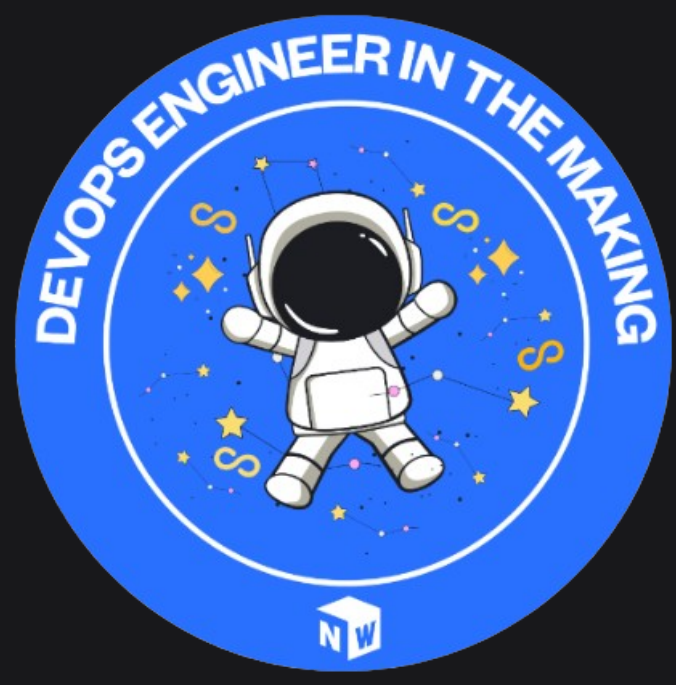
Save for later

Let's connect!

Dahri Hadri

@dahrihadri

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



Ask me about it