



NextWork.org

Deploy an App with CodeDeploy



Kanika Mathur
github.com/KanikaGenesis



Not secure

44.200.9.204

Hello Kanika!

This is my NextWork web application working!



Kanika Mathur
github.com/KanikaGenesis

NextWork.org

Introducing AWS CodeDeploy!

What it does & how it's useful

AWS CodeDeploy is a service that automates code deployments to any instance, including Amazon EC2 and on-premises servers. Developers and teams use AWS CodeDeploy because it simplifies deployment, reduces downtime, and integrates easily with existing CI/CD tools, ensuring faster and more reliable releases.

How I'm using it in today's project

I am using AWS CodeDeploy in this project to deploy the WAR file (that we created using CodeBuild) on servers, so users can see my web app!

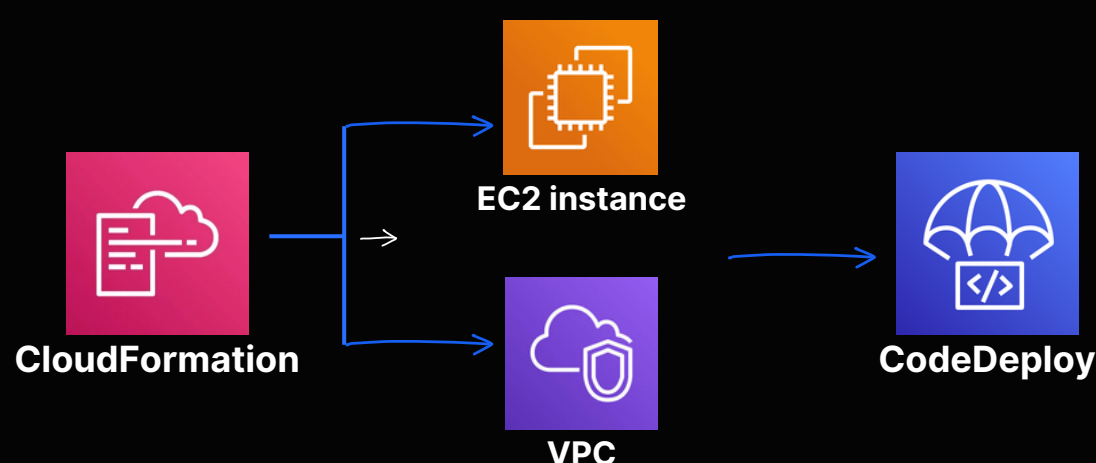
This project took me...

The project took me around 5 hours to complete with documentation.



Set up an EC2 instance

- I set up an EC2 instance and VPC because deploying my web app requires a separate environment (my production environment).
- The production environment is different to the build environment because deploying my web app will require different tools/dependencies in order to function properly. It is also best practice to separate environments for developers to test code without interfering with what users see.
- To set up my EC2 instance and VPC, I used AWS CloudFormation.
- The diagram below illustrates the flow of using CloudFormation to create two new resources (EC2 instance, VPC) that will run/deploy my web app (which we will see in action when we set up AWS CodeDeploy).



A peek into my CloudFormation stack's deployment!

The screenshot shows the AWS CloudFormation Events console for a stack with 34 events. A blue arrow points to the 'Events (34)' header. The table below lists the events, showing the creation of PublicInternetRoute and PublicRouteTable resources.


Timestamp	Logical ID	Status	Detailed status	Status reason
2024-06-20 11:41:06 UTC-0230	PublicInternetRoute	✔ CREATE_COMPLETE	-	-
2024-06-20 11:41:06 UTC-0230	PublicInternetRoute	ℹ CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-06-20 11:41:05 UTC-0230	PublicInternetRoute	ℹ CREATE_IN_PROGRESS	-	-
2024-06-20 11:41:04 UTC-0230	PublicRouteTable	✔ CREATE_COMPLETE	-	-



Write bash scripts

- Scripts are collections of commands in a file i.e when you run a script, you are telling the terminal to run (line by line) what you've written in the script.
- Bash is a specific type of script language.
- I created 3 scripts for this the deployment process:
 - install_dependencies.sh installs Tomcat and HTTPD (web servers)
 - start_server.sh starts up our web servers
 - stop_server.sh stops our web servers

A peek into appspec.yml!



```
1  version: 0.0
2  os: linux
3  files:
4    - source: /target/nextwork-web-project.war
5      destination: /usr/share/tomcat/webapps/
6  hooks:
7    BeforeInstall:
8      - location: scripts/install_dependencies.sh
9        timeout: 300
10       runas: root
11    ApplicationStart:
12      - location: scripts/start_server.sh
13        timeout: 300
14       runas: root
15    ApplicationStop:
16      - location: scripts/stop_server.sh
17        timeout: 300
18       runas: root
```



CodeDeploy's IAM Role

- I created an IAM service role for CodeDeploy because CodeDeploy needs access to other AWS services e.g. EC2 in order to successfully deploy web app.
- To set up CodeDeploy IAM role, I used an AWS Managed Policy called **AWSCodeDeployRole** which automatically adds default permissions that CodeDeploy often needs.

The permissions granted by my CodeDeploy IAM role

Policy name

[AWSCodeDeployRole](#)

AWSCodeDeployRole

Provides CodeDeploy service access to expand tags and interact with Auto Scaling on your behalf.

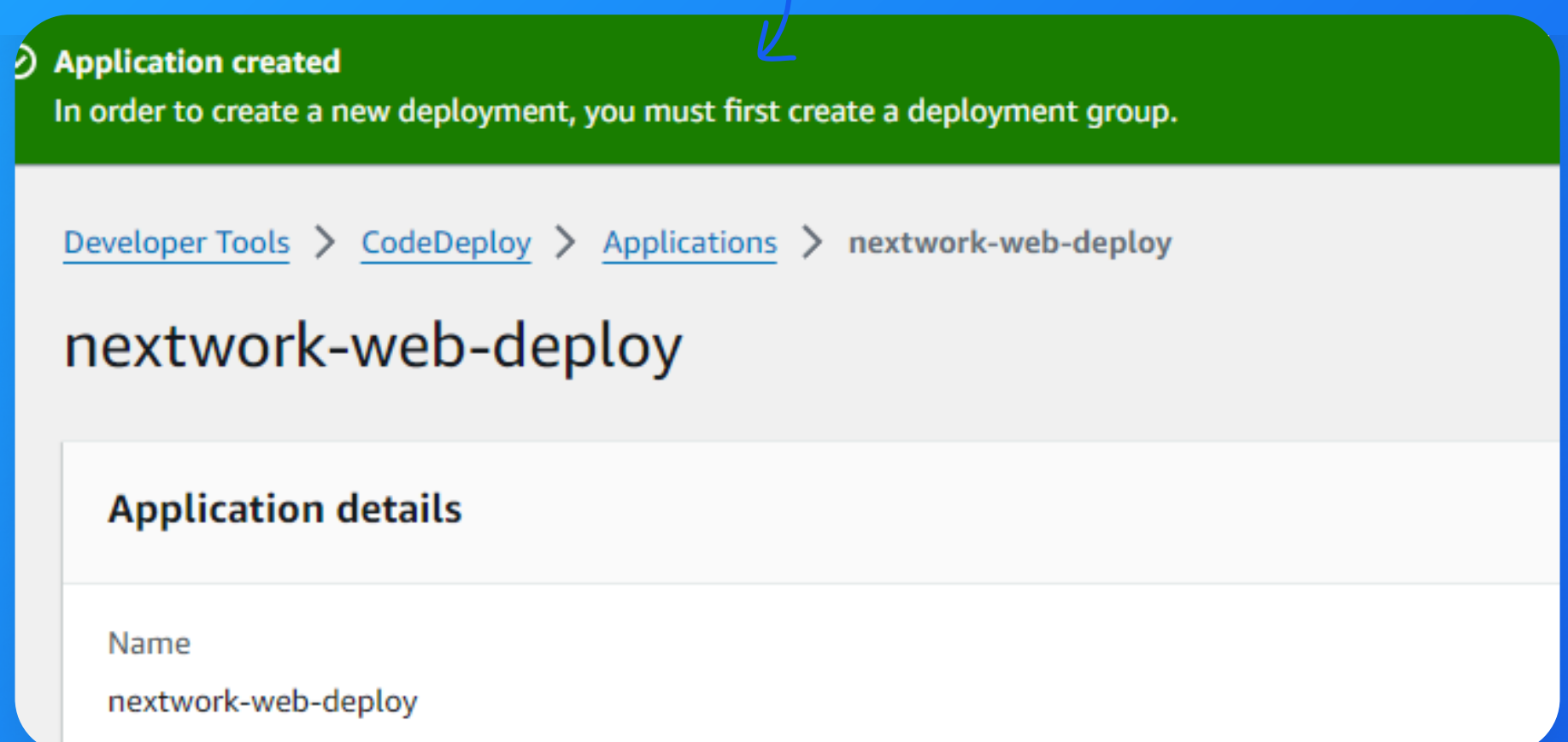
```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Effect": "Allow",
6       "Action": [
7         "autoscaling:CompleteLifecycleAction",
8         "autoscaling>DeleteLifecycleHook",
9         "autoscaling:DescribeAutoScalingGroups",
10        "autoscaling:DescribeLifecycleHooks",
11        "autoscaling:PutLifecycleHook",
12        "autoscaling:RecordLifecycleActionHeartbeat",
13        "autoscaling>CreateAutoScalingGroup",
14        "autoscaling>CreateOrUpdateTags",
15        "autoscaling:UpdateAutoScalingGroup",
```



Create a CodeDeploy app

- A CodeDeploy application means a saved configuration/setup template on how to deploy my web app.
- To create a CodeDeploy application, I had to select a compute platform, which determines the environment in which my web application will be hosted. With every platform comes its own set of requirement for deploying a web app.
- The compute platform I chose was EC2, because I have used an EC2 instance as my web server. Choosing EC2 gives me the most amount of control out of all the three compute platform options. It's also great for my learning as I will be exposed to more configuration options!

My CodeDeploy app ready for a deployment!



The screenshot shows the AWS CodeDeploy console. At the top, a green banner with a checkmark icon says "Application created" and "In order to create a new deployment, you must first create a deployment group." Below this is a breadcrumb trail: "Developer Tools > CodeDeploy > Applications > nextwork-web-deploy". The main heading is "nextwork-web-deploy". Underneath, there's a section titled "Application details" with a table containing one row: "Name" with the value "nextwork-web-deploy". A blue arrow from the text above points to the green banner.

Application created
In order to create a new deployment, you must first create a deployment group.

[Developer Tools](#) > [CodeDeploy](#) > [Applications](#) > nextwork-web-deploy

nextwork-web-deploy

Application details	
Name	nextwork-web-deploy



Set up a deployment group

- A deployment group means the configuration set for a specific deployment scenario.
- To create my deployment group, I set up a:
 - **Service role**, which is an IAM role that I am using for my deployment group to give CodeDeploy access to my Web Server EC2 instance.
 - **Deployment type**, which is how deployment will be managed. I chose In place, i.e. my Web Server EC2 instance will deploy the latest web app right away without needing to create a new environment.
 - **Environment configuration**, which means the type of servers that will be used to deploy my web app - in this case, I just used Amazon EC2 instances (without needing Auto Scaling groups).
 - **CodeDeploy Agent**, which is CodeDeploy's helper for communicating with my EC2 Web Server and making sure that carry out the instructions for deployment.
 - **Deployment settings**, which are whether deployment will be staggered out or done all out once.
- For the load balancer setting, I disabled the load balancer.
 - This was because I only have one web server deploying my web app.



Setting up the service role + deployment type

Service role

Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.

Deployment type

Choose how to deploy your application

☒ **In-place**
Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

☐ **Blue/green**
Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Setting up the service role + deployment type

Agent configuration with AWS Systems Manager [Info](#)

Complete the required prerequisites before AWS Systems Manager can install the CodeDeploy Agent.
Make sure the AWS Systems Manager Agent is installed on all instances and attach the required IAM policies to them. [Learn more](#)

Install AWS CodeDeploy Agent

☐ Never
☐ Only once
☒ Now and schedule updates

Basic scheduler Cron expression

Days



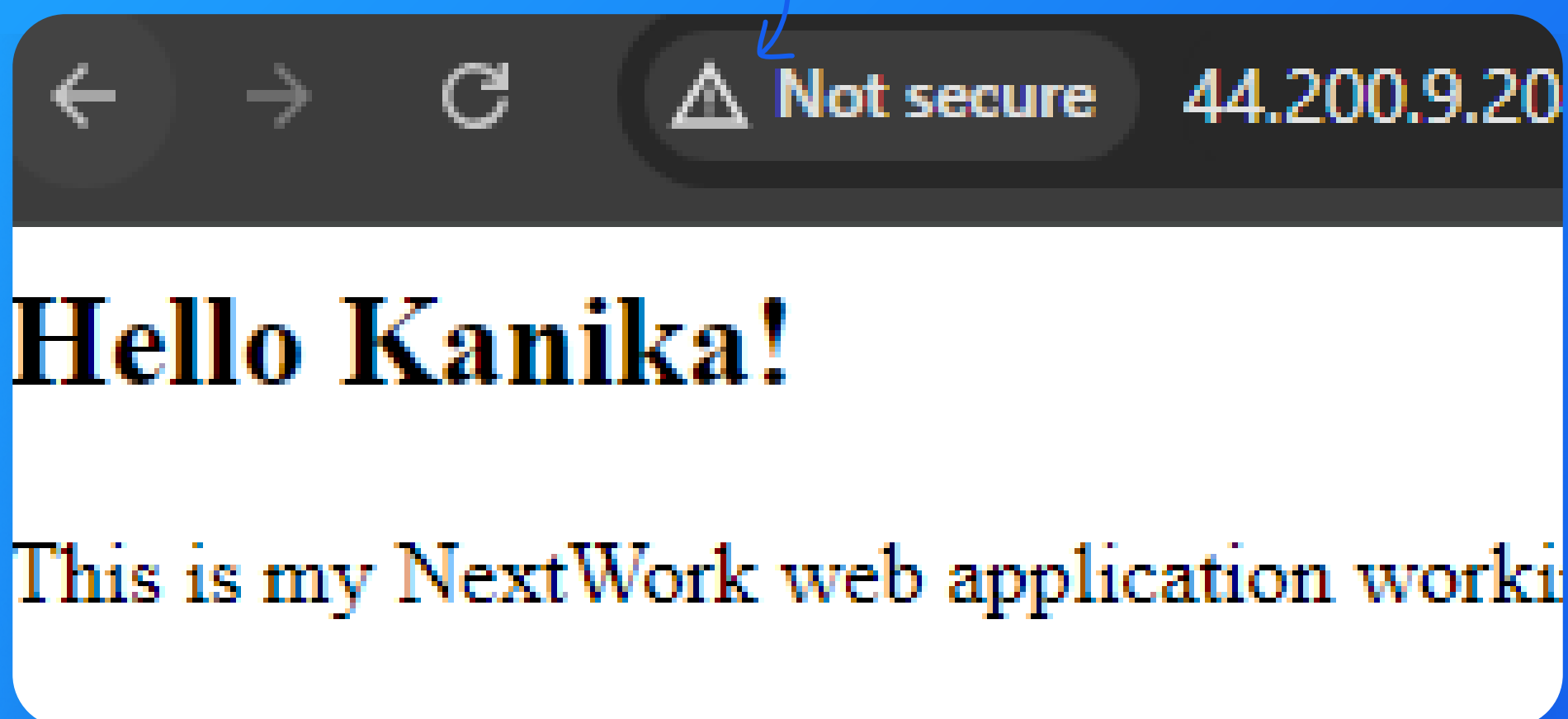
Kanika Mathur
github.com/KanikaGenesis

NextWork.org

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 the location of my web app's build artifacts.
- My revision location was the Java WAR file/zip file containing my compiled web app code. It was sitting inside my S3 build artifacts bucket.
- To visit my web app, I had to visit my Web Server which was in my EC2 console. I have to find my EC2 instance's IPv4 address.

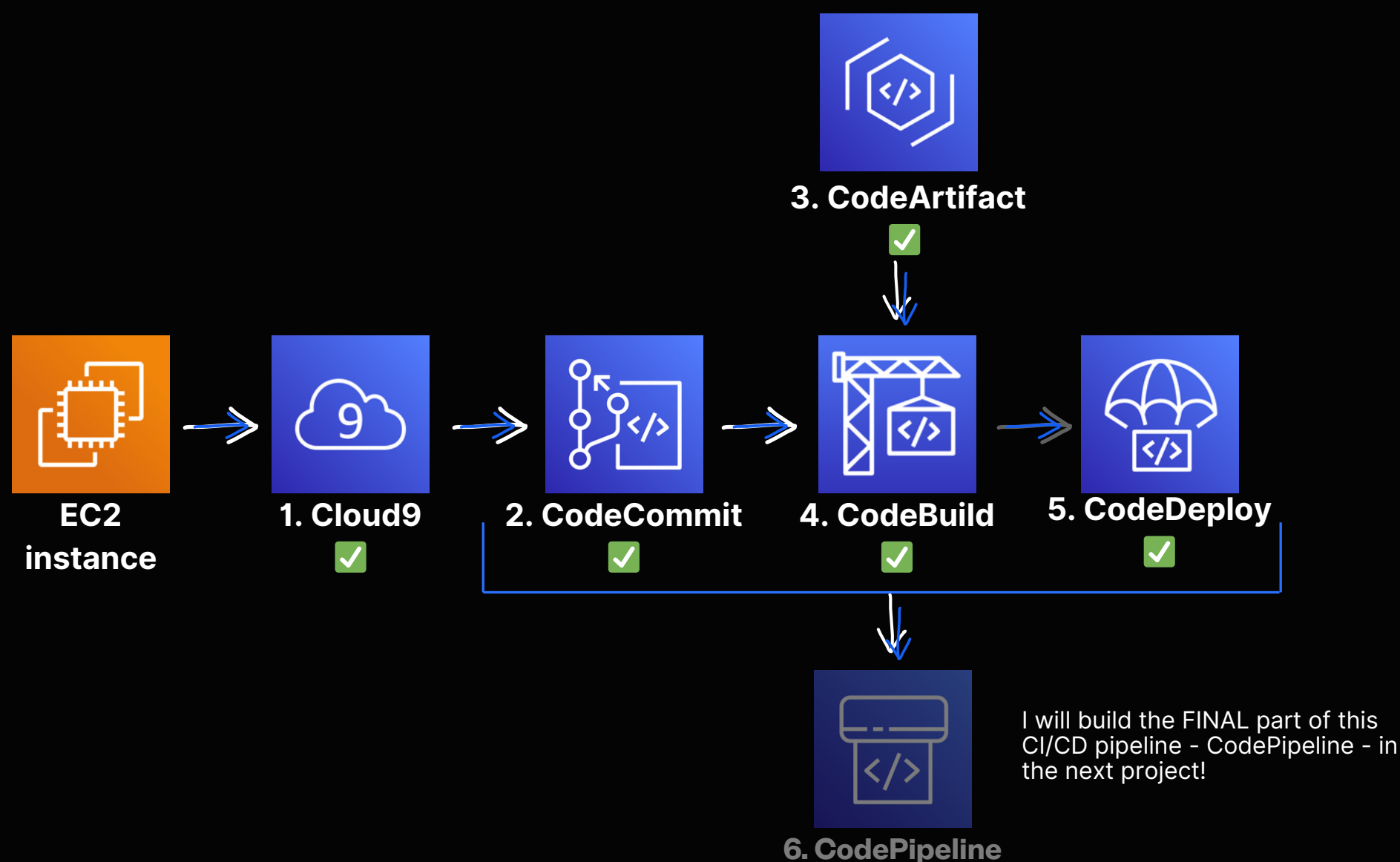
Amazing! I could see my web app taking shape.





My CI/CD pipeline so far...

- AWS Cloud9 is responsible for providing a cloud-based IDE where I can write, run, and debug my code.
- AWS CodeCommit is responsible for securely hosting my Git repositories, enabling version control and collaboration.
- AWS CodeArtifact is responsible for managing and storing software packages, making it easy to share dependencies across teams.
- AWS CodeBuild is responsible for compiling my source code, running tests, and producing build artifacts.
- AWS CodeDeploy is responsible for automating the deployment of my applications to various compute services, ensuring smooth and efficient updates.





My key learnings

- 1 The deployment process means automating the release of applications or updates to production or other environments. This ensures that new code changes are reliably and consistently delivered with minimal downtime and without manual intervention
- 2 I created a separate environment for deployment because it allows for thorough testing and validation of changes in a controlled setting before they go live. This helps ensure stability, minimizes risks, and prevents disruptions in the production environment.
- 3 To create a deployment, I had to set up different resources in AWS, which included: an EC2 instance, scripts to run the application, CodeDeploy service IAM Role with necessary permissions, a CodeDeploy application and a deployment group.
- 4 One thing I loved doing this project was how smoothly the deployment process went once everything was set up correctly. It was rewarding to see the automated pipeline work seamlessly.



Everyone should be in a job they love. *yes!*

Check out community.nextwork.org for more free projects



Ask me about it

www.linkedin.com/in/kanika-mathur-083080121

