



NextWork.org

CLOUD DEVOPS: FROM DEVELOPMENT TO DEPLOYMENT



Kanika Mathur
github.com/KanikaGenesis



**EVERYONE SHOULD BE IN
A JOB THEY LOVE** 



You?

Building the best online learning experience to switch careers and upskill.

Starting with AWS certifications.

Enter your email

Join waitlist

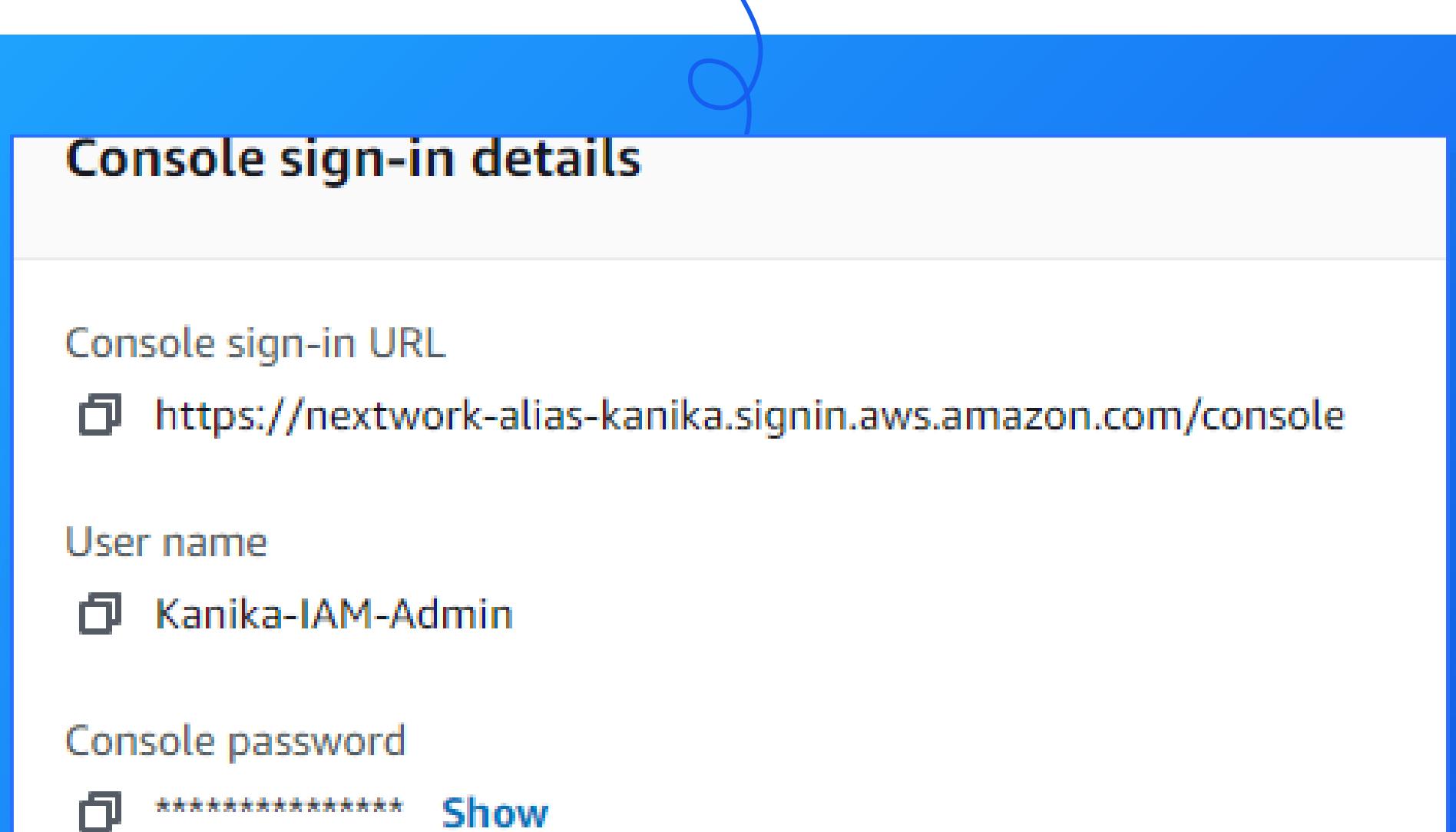


Set up an IAM User

An IAM user is an additional user that gets access to my AWS Account's resources. When creating a User, I can specify in detail the level of access it has to my account's resources and services.

- It's important to create IAM users because the root user is vulnerable to security breaches that could result in my billing information being accessed without my permission.
- I created an IAM user with Administrator Access. This means my IAM user is allowed to perform all possible actions to all the resources in my account.

A new IAM user set up for my AWS Account



The screenshot shows the 'Console sign-in details' section of the AWS IAM user configuration. It includes the console sign-in URL (<https://nextwork-alias-kanika.signin.aws.amazon.com/console>), user name (Kanika-IAM-Admin), and a password field (*****) with a 'Show' link.

Console sign-in details

Console sign-in URL

□ <https://nextwork-alias-kanika.signin.aws.amazon.com/console>

User name

□ Kanika-IAM-Admin

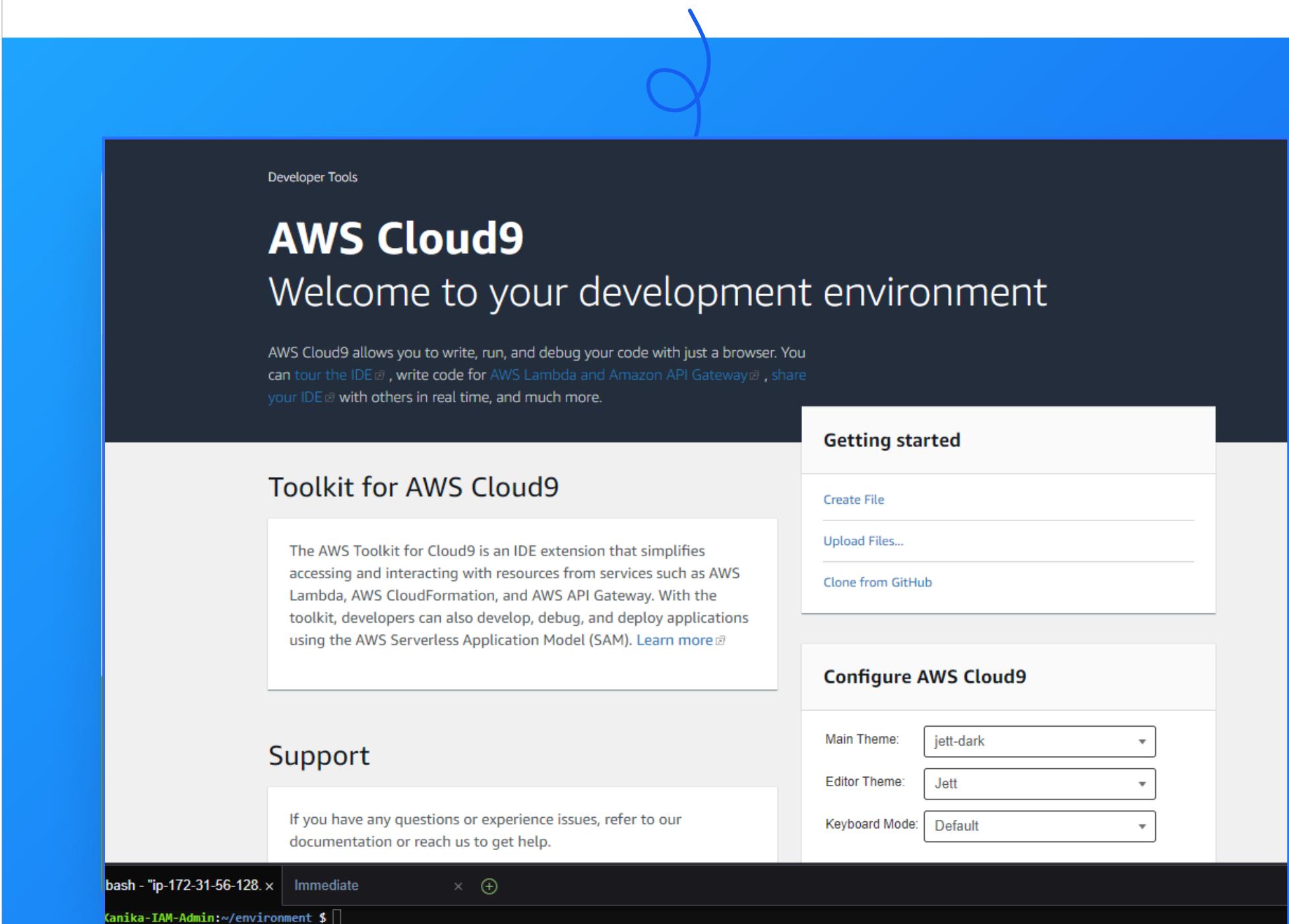
Console password

□ ***** [Show](#)

Launch a Cloud9 IDE

- An IDE is an environment for developers to write, manage and debug their code. Just like how Google Docs is a companion of writers to create documents, IDEs are companions for developers to build their application/ edit their code.
- I used AWS Cloud9 to launch an environment. An environment means a set of resources that work together to help me build a piece of software/ application. Every application has its own needs e.g. dependencies, libraries, tools, etc. This means the environment that I'm creating will be tailored to the web application I build.
- Using Cloud9 meant that I did not need to download this software in order to use an IDE.

My Cloud9 IDE!





Install Maven & Java

- Maven is a tool that helps developers with the building of their software, automating the steps required for my application to become a final product that machines e.g. computers can actually run.
- Maven is required in this project because I am building a web app in a specific programming language which will not be able to build on its own without the help of Maven.
- Java is the specific programming language I am using to create my web application.
- Java is required in this project because it is a versatile tool for creating different applications, including web apps.
- The Java version I'm using for this project is Amazon Coretto 8.

I used terminal commands to install Maven and Java



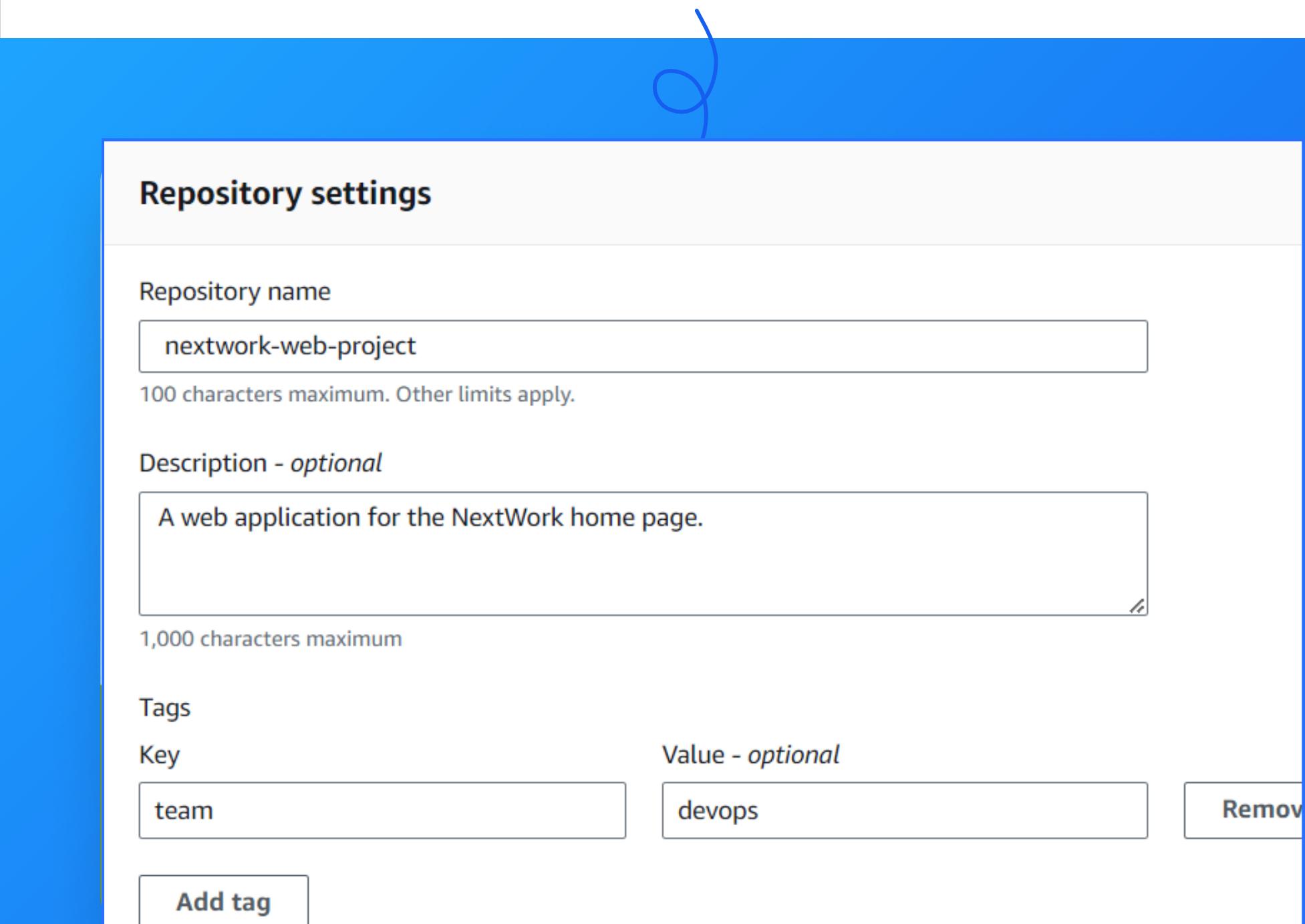
```
Dependency Installed:  
java-1.8.0-amazon-corretto.x86_64 1:1.8.0_412.b08-1.amzn2  
  
Complete!  
Kanika-IAM-Admin:~/environment $ export JAVA_HOME=/usr/lib/jvm/java-1.8.0-amazon-corretto.x86_64  
Kanika-IAM-Admin:~/environment $ export PATH=/usr/lib/jvm/java-1.8.0-amazon-corretto.x86_64/jre/bin/:$PATH  
  
Kanika-IAM-Admin:~/environment $ java -version  
openjdk version "1.8.0_412"  
OpenJDK Runtime Environment Corretto-8.412.08.1 (build 1.8.0_412-b08)  
OpenJDK 64-Bit Server VM Corretto-8.412.08.1 (build 25.412-b08, mixed mode)  
Kanika-IAM-Admin:~/environment $ mvn -v  
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z)
```



Create a Git repository

- Git is a version control and code management system that helps developers to tracking their changes and collaborating on code together.
- A Git repository is like a folder that contain all of the application/project's file in one place.
- To create a Git repository in the cloud, I used CodeCommit.

My setup page for a CodeCommit repo



The screenshot shows the 'Repository settings' page for a new CodeCommit repository. The repository name is 'nextwork-web-project'. The description is 'A web application for the NextWork home page.' There is one tag named 'team' with value 'devops'. A green bar at the bottom indicates the form is valid.

Repository settings

Repository name

nextwork-web-project

100 characters maximum. Other limits apply.

Description - *optional*

A web application for the NextWork home page.

Tags

Key	Value - <i>optional</i>	Remove
team	devops	Remove

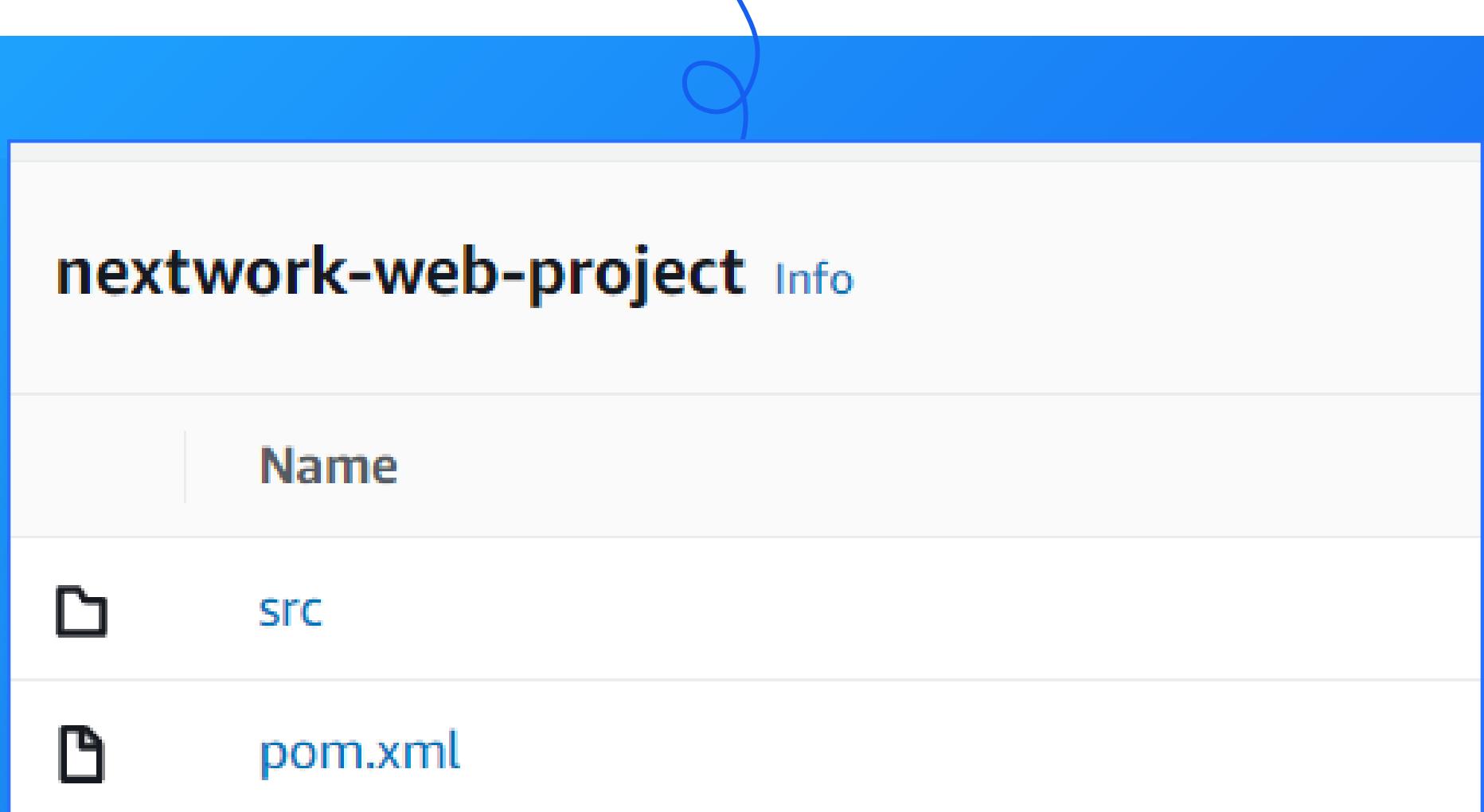
Add tag



My first commit

- I initialised a Git repo in my web application by running the command **git init -b main**.
- To commit and push my code, I will have to run three different commands in order:
- **git add** places the file that I've created/edited in a staging area i,e. preparing them to be saved.
- **git commit** is like pushing a “save” button that confirms my changes.
- **git push** sends my changes upstream to my remote origin i,e. CodeCommit repository I've set up.

Files I committed showing up in my CodeCommit repo!



The screenshot shows the AWS CodeCommit interface. At the top, there's a blue header bar with a search icon. Below it, the repository name "nextwork-web-project" is displayed in large, bold, blue text, followed by a "Info" link. A large blue curly brace icon is positioned above the file list. The file list table has a single column labeled "Name". Underneath the table, there are two items: "src" with a folder icon and "pom.xml" with a document icon.

Name
src
pom.xml



Git in action

- I wanted to see Git working in action, so I updated my index.jsp file by adding two new lines.
- Then I tried seeing these changes in my CodeCommit repository, but this didn't work because I had only saved these changes in my local repository without pushing the changes upstream.
- I finally saw the changes in my CodeCommit repository after running the same three Git commands in my Cloud9 terminal:
 - git add .
 - git commit
 - git push

My updated index.jsp file showing up in CodeCommit!



The screenshot shows a terminal window with a blue header bar. The header bar has a small circular icon with a question mark and the text "nextwork-web-project / src / main / webapp / index.jsp". Below the header, the terminal displays the following code:

```
1 <html>
2
3 <body>
4
5 <h2>Hello Kanika!</h2>
6
7 <p>This is my NextWork web application working!</p>
8 <p>Yo! If you see this line in CodeCommit, your latest changes are saved in th
9
10 </body>
11
12 </html>
```

Create a repository

- CodeArtifact is a service that sits in the CI/CD pipeline. I am using it today to store backup copies of packages (importantly, dependencies) relevant to my java web app.
- The reason why I'm using CodeArtifact for my web app is for security/risk management and continuity. Even if public packages or dependencies of my project are no longer available, there is a copy in my AWS CodeArtifact repositories to make sure I can keep developing my web app.
- Instead of a single repository, there are actually three connected repositories that Maven uses to fetch packages.
- The first is my local repository, which Maven checks for packages/dependencies for my web app.
- The second is my public upstream repository which Maven will check next if the package is not in the local repository.
- The third is the Maven Central Repository, which is a public repository with the greatest collection of packages for Java application. However, Maven will only visit this repository if the first two do not have the packages/ dependencies it is looking for (due to high traffic going into Maven Central Repository).

Package flow illustrating the connections between the three repositories.



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

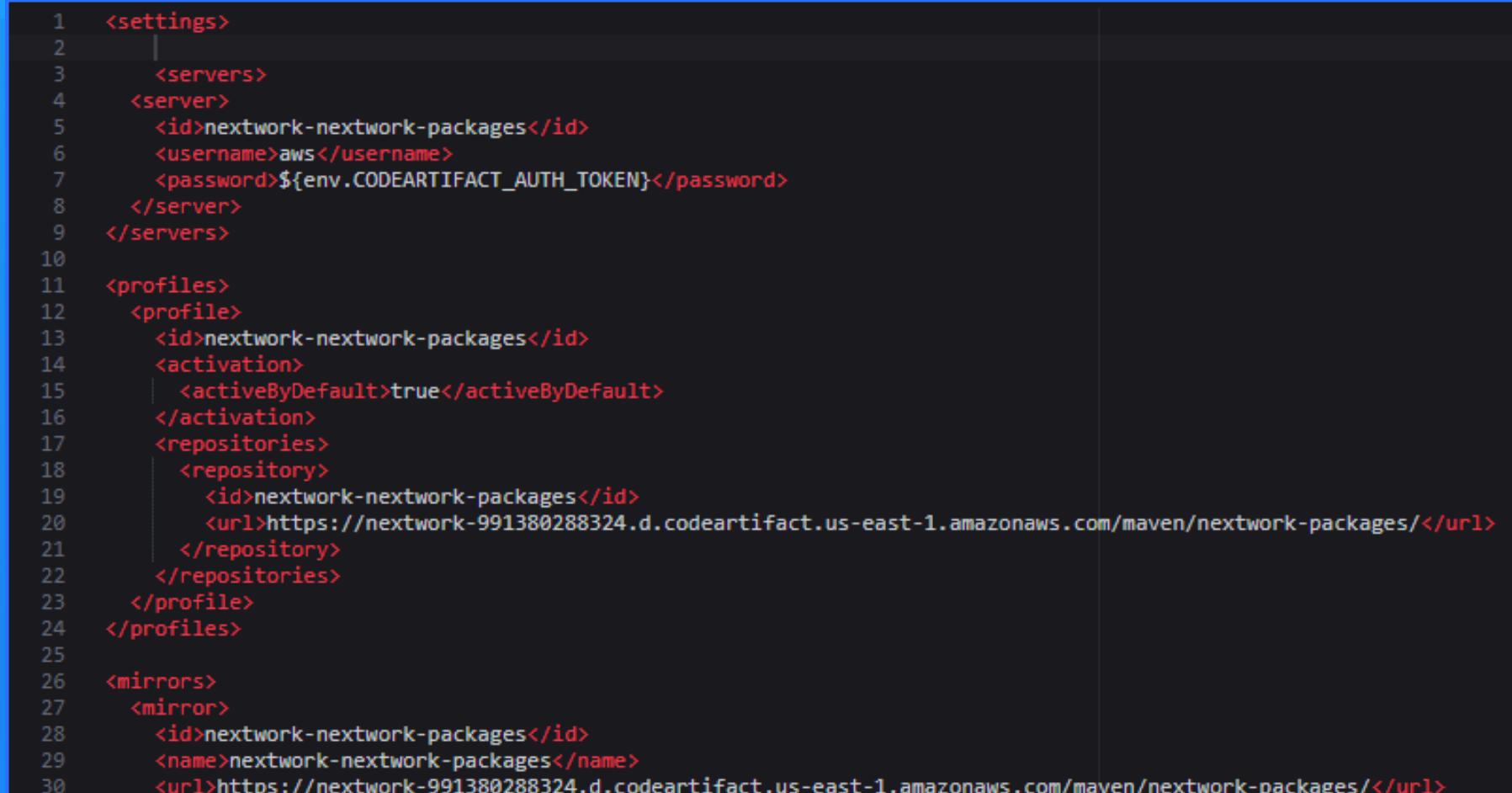
Repository
nextwork



Connecting my project to CodeArtifact

- Next, I connected my web app project (via my Cloud9 IDE) to CodeArtifact so that CodeArtifact knows which project it is going to storing dependencies for!
- I created a new file, settings.xml, in my web app. settings.xml is the file that will give Maven instructions on WHERE to find the dependencies Maven will need to fetch, and HOW Maven will get access to these repositories that are storing these dependencies.
- 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 when Maven will visit which repository, plus where Maven should visit to find backup local repositories (optional).

My settings.xml file.



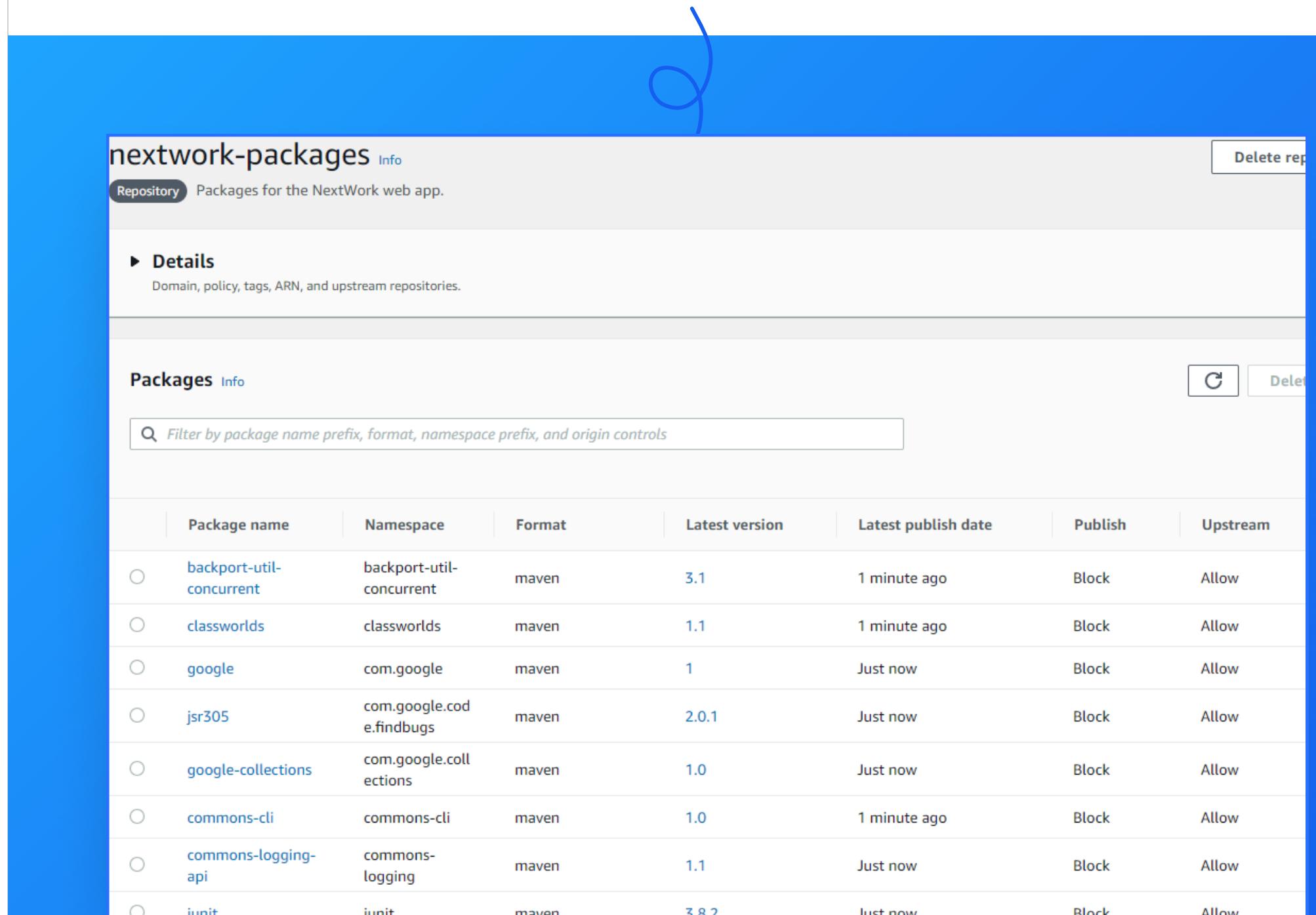
```
1 <settings>
2   |
3     <servers>
4       <server>
5         <id>nextwork-nextwork-packages</id>
6         <username>aws</username>
7         <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
8       </server>
9     </servers>
10
11   <profiles>
12     <profile>
13       <id>nextwork-nextwork-packages</id>
14       <activation>
15         <activeByDefault>true</activeByDefault>
16       </activation>
17       <repositories>
18         <repository>
19           <id>nextwork-nextwork-packages</id>
20           <url>https://nextwork-991380288324.d.codeartifact.us-east-1.amazonaws.com/maven/nextwork-packages/</url>
21         </repository>
22       </repositories>
23     </profile>
24   </profiles>
25
26   <mirrors>
27     <mirror>
28       <id>nextwork-nextwork-packages</id>
29       <name>nextwork-nextwork-packages</name>
30       <url>https://nextwork-991380288324.d.codeartifact.us-east-1.amazonaws.com/maven/nextwork-packages/</url>
```



Test the connection

- To test the connection between Cloud9 and CodeArtifact, I compiled my web app. Compiling means translating my web application's code into machine code that servers can actually understand and run.
- After compiling, I checked my local repository and saw that my local repository now has pages and pages of packages inside! This means Maven has now grabbed packages from the upstream repository/Maven Central Repository and installed/ kept a copy locally.

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



The screenshot shows a user interface for managing a local repository. At the top, there is a blue header bar with a logo on the left and a 'Delete rep' button on the right. Below the header, the repository name 'nextwork-packages' is displayed along with an 'Info' link. A 'Repository' tab is selected, showing the message 'Packages for the NextWork web app.' Below this, a 'Details' section is shown with a sub-section for 'Domain, policy, tags, ARN, and upstream repositories.' Further down, a 'Packages' section is visible, featuring a search bar with the placeholder 'Filter by package name prefix, format, namespace prefix, and origin controls'. A table lists several Maven packages:

	Package name	Namespace	Format	Latest version	Latest publish date	Publish	Upstream
○	backport-util-concurrent	backport-util-concurrent	maven	3.1	1 minute ago	Block	Allow
○	classworlds	classworlds	maven	1.1	1 minute ago	Block	Allow
○	google	com.google	maven	1	Just now	Block	Allow
○	jsr305	com.google.code.findbugs	maven	2.0.1	Just now	Block	Allow
○	google-collections	com.google.collections	maven	1.0	Just now	Block	Allow
○	commons-cli	commons-cli	maven	1.0	1 minute ago	Block	Allow
○	commons-logging-api	commons-logging	maven	1.1	Just now	Block	Allow
○	junit	junit	maven	3.8.2	Just now	Block	Allow



Create IAM policies

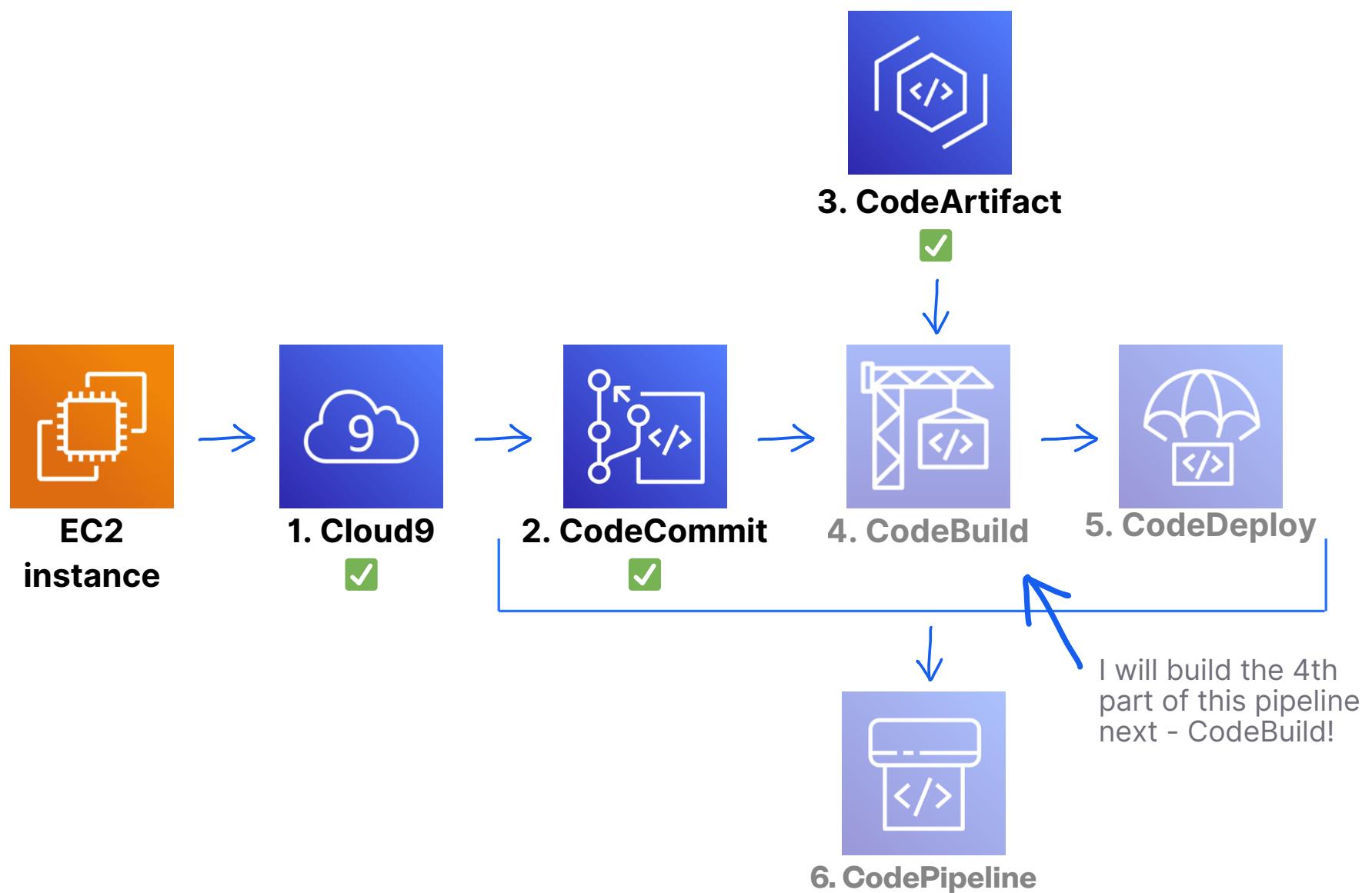
- I also created an IAM policy because other services in my CI/CD pipeline e.g. CodeBuild, CodePipeline will be needing access to the packages/dependencies stored in CodeArtifact. By default, these services do NOT have access - so they need to be granted access through an IAM Policy.
- I defined my IAM policy using JSON. This policy will enable the policy holder to get authorization token (i.e. access to CodeArtifact), fetch packages stored in CodeArtifact's repositories.

A peek at the policy that will provide access to CodeArtifact

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

My CI/CD pipeline so far...

1. Cloud9 is responsible for IDE that allows you to write ,run and debug your code.
2. CodeCommit is responsible for hosting git repositories.
3. CodeArtifact is responsible for securely storing and sharing packages/dependencies used in software development process.

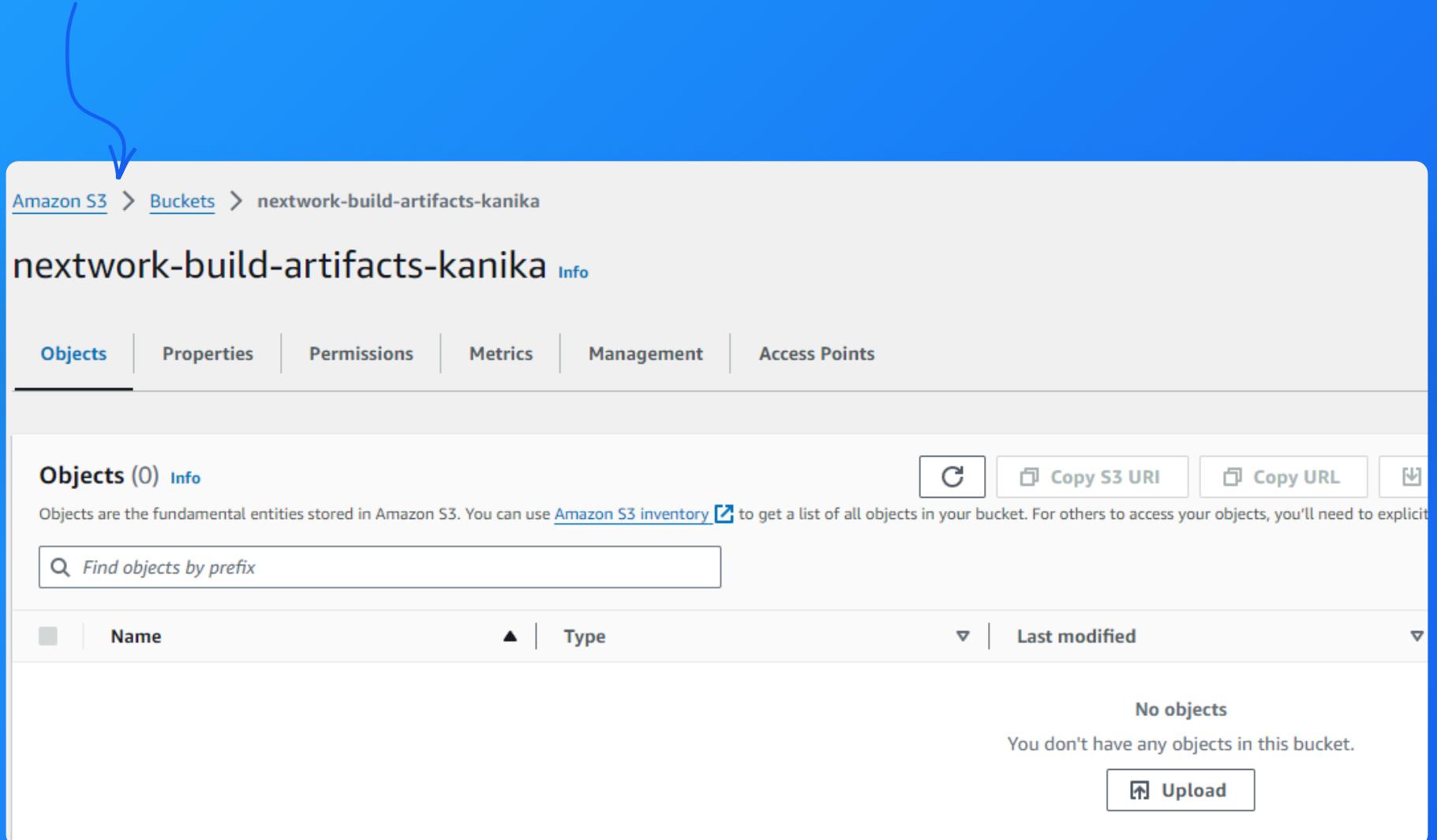




Set up an S3 bucket

- I started my project by creating an S3 bucket because this bucket will later catch an important artifact that gets created from the build process I am setting up with CodeBuild.
- The key artifact that this S3 bucket will capture is called a WAR (Web Application Resource) file.
- This file is important because it ensures that any server that will host my web app will have all of the resources/tools it needs to successfully run my application.

My S3 bucket!





Kanika Mathur
github.com/KanikaGenesis

NextWork.org

Set up a CodeBuild project

When creating a project in CodeBuild, there were 5 key configurations I set up:

1. **Source**, which means the source of the project's code/files. I selected AWS CodeCommit, as that is my cloud repository for my project's files.
2. **Environment**, which means the set of resources/configurations required to build my web app. I selected an EC2 instance running Amazon Linux 2 and Java Coretto 8 as the runtime.
3. **Buildspec**, which are the set of commands to run in my local computer's terminal during the build process. I selected "use a buildspec file."
4. **Artifacts**, which means where CodeBuild should store the build artifacts that get produced during the build process. I selected the S3 bucket I created.
5. **Logs**, which means whether I'd like to keep records of every single build process that CodeBuild does. I enabled CloudWatch Logs.

My completed project ready for the first build!

The screenshot shows the AWS CodeBuild console interface. At the top, there is a navigation bar with the project name 'nextwork-web-build' and several action buttons: Actions ▾, Create trigger, Edit, Clone, Debug build, Start build with overrides, and Start build. Below the navigation bar is a 'Configuration' section containing four main fields: 'Source provider' (AWS CodeCommit), 'Primary repository' (nextwork-web-project), 'Artifacts upload location' (nextwork-build-artifacts-kanika), and 'Service role' (arn:aws:iam::991380288324:role/service-role/codebuild-nextwork-web-build-service-role). Underneath the configuration is a 'Build history' tab, which is currently selected. This tab has several sub-tabs: Build history, Batch history, Project details, Build triggers, and Metrics. Below these tabs is a 'Build history' table header with columns: Build run, Status, Build number, Source version, Submitter, Duration, and Completed. The table body displays a single row with the status 'No results' and the message 'There are no results to display.'

Create a buildspec.yml file

- I created a buildspec.yml file at the root of our code repository.
- This file contains four phases that tells our build environment what commands to run. These four phases are:
 1. install - i.e. install these dependencies before you start compiling (in this project, it's Java Coretto 8).
 2. pre_build - i.e. before Maven starts building retrieve an access token to our CodeArtifact repository so we can fetch dependencies later.
 3. build - i.e. to build our web app project, run these commands.
 4. post_build - i.e. after building our web app project, package up the machine code using settings in the setting.xml file.

A peek into my buildspec.yml

```
1  version: 0.2
2
3  phases:
4    install:
5      runtime-versions:
6        java: corretto8
7    pre_build:
8      commands:
9        - echo Initializing environment
10       - export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain ne
11    build:
12      commands:
13        - echo Build started on `date`
14        - mvn -s settings.xml compile
15    post_build:
16      commands:
17        - echo Build completed on `date`
18        - mvn -s settings.xml package
19    artifacts:
20      files:
21        - target/nextwork-web-project.war
22    discard-paths: no
```

Kanika Mathur
github.com/KanikaGenesis

NextWork.org

Edit CodeBuild's IAM role

- Before I start building my web app project (exciting!), I modified my CodeBuild project's service role first. This role was first created when I set up my CodeBuild project (I checked the setting for create new service role).
- I attached a new policy called **codeartifact-nextwork-consumer-policy** to my CodeBuild project's IAM role. This means my CodeBuild project now has access to the packages/dependencies that it will later compile.

Updating permission policies for my CodeBuild project's IAM role.

A screenshot of the AWS IAM Permissions Policies page. The page title is "Permissions policies (3) Info". A blue callout bubble points to the "Info" link. Below the title, a message says "You can attach up to 10 managed policies." A search bar is at the top. The main list shows three policies:

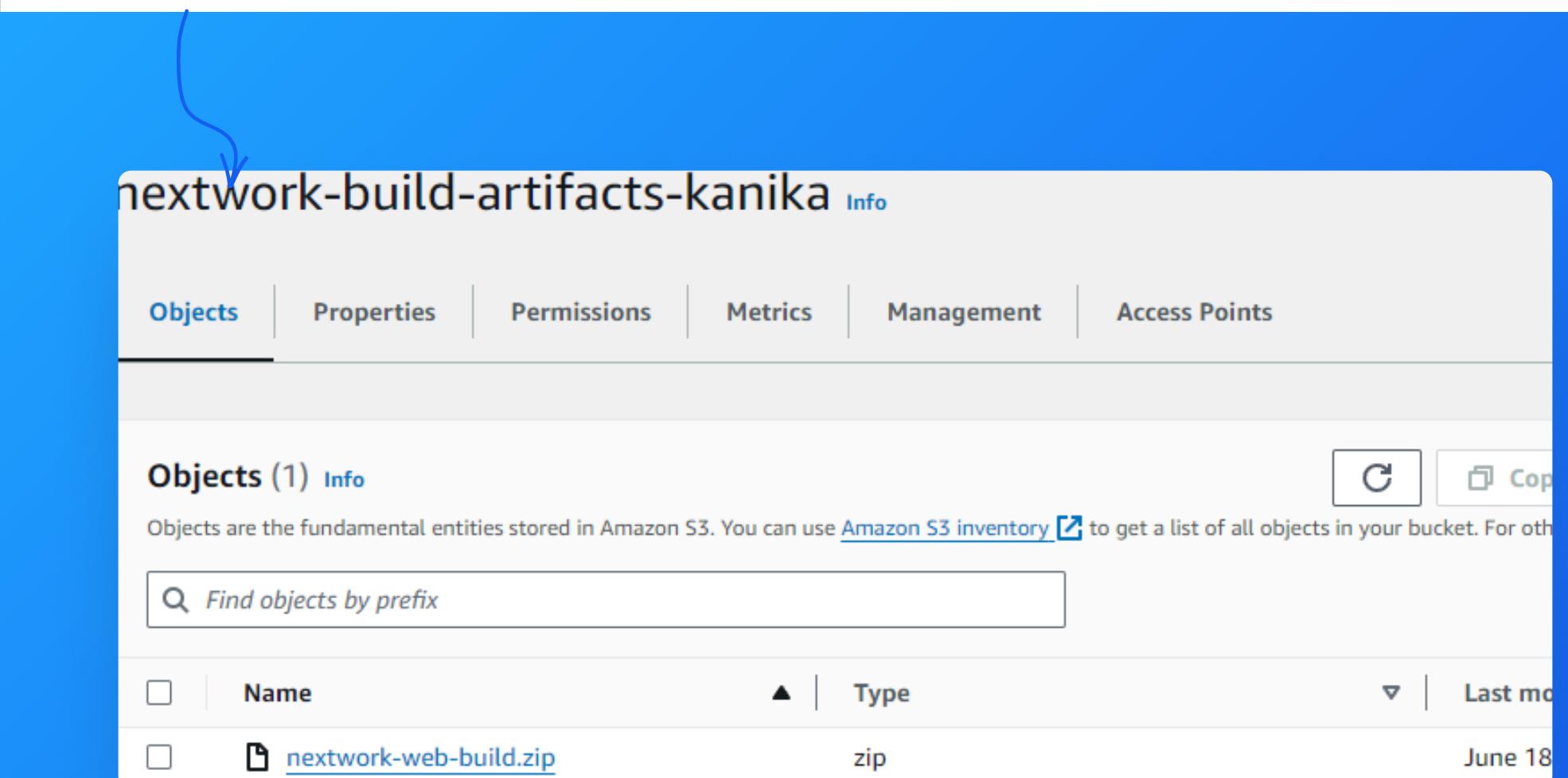
Policy name
<input type="checkbox"/> codeartifact-nextwork-consumer-policy
<input type="checkbox"/> CodeBuildBasePolicy-nextwork-web-build-us-east-1
<input type="checkbox"/> CodeBuildCloudWatchLogsPolicy-nextwork-web-build-us-ea



My first project build 💪

- To build my project, all I had to do was select the Start build button in my CodeBuild console - super simple!
- The build process in CodeBuild took about 5 minutes.
- Once the build is complete, I checked our S3 bucket was set up at the start of this project to catch the build artifacts that get produced from our build process. I saw the zip file i.e. the WAR file (a bundle up package of all files/resources a server will need to run my web app), which verified that the build was completed successfully.

My completed project ready for the first build!



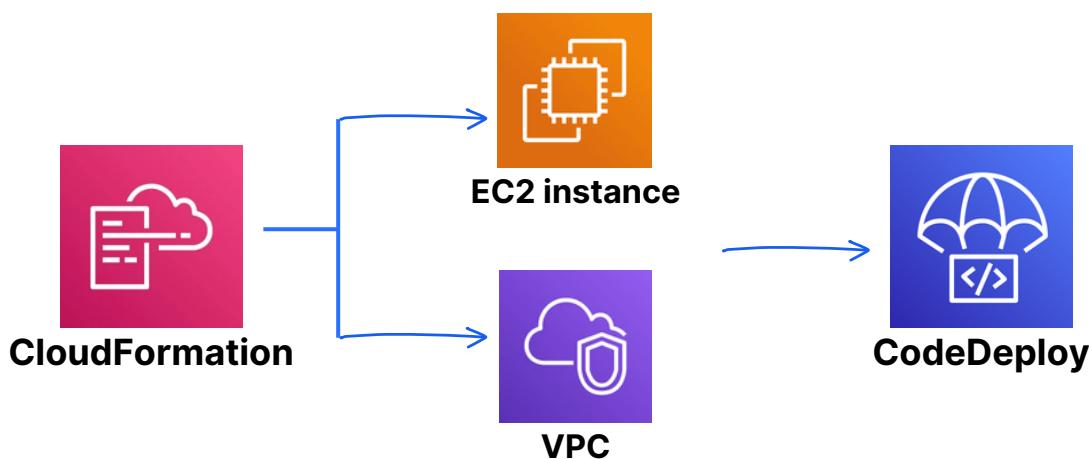
The screenshot shows the Amazon S3 console interface. At the top, there's a blue header bar with a navigation icon and the text "nextwork-build-artifacts-kanika". Below the header, there are tabs for "Objects", "Properties", "Permissions", "Metrics", "Management", and "Access Points". The "Objects" tab is selected, showing a table with one item: "nextwork-web-build.zip". The table has columns for "Name" and "Type". A blue arrow points from the text "My completed project ready for the first build!" to the bucket name "nextwork-build-artifacts-kanika".

Name	Type	Last modified
nextwork-web-build.zip	zip	June 18



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!

Events (34)				
<input type="text"/> Search events				
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.

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.

The screenshot shows the AWS IAM Policies page with a blue header. Below the header, there is a search bar with a magnifying glass icon and a dropdown menu labeled "AWS CodeDeploy". The main content area displays a policy named "AWSCodeDeployRole".

Policy name

 [AWSCodeDeployRole](#)

AWSCodeDeployRole

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

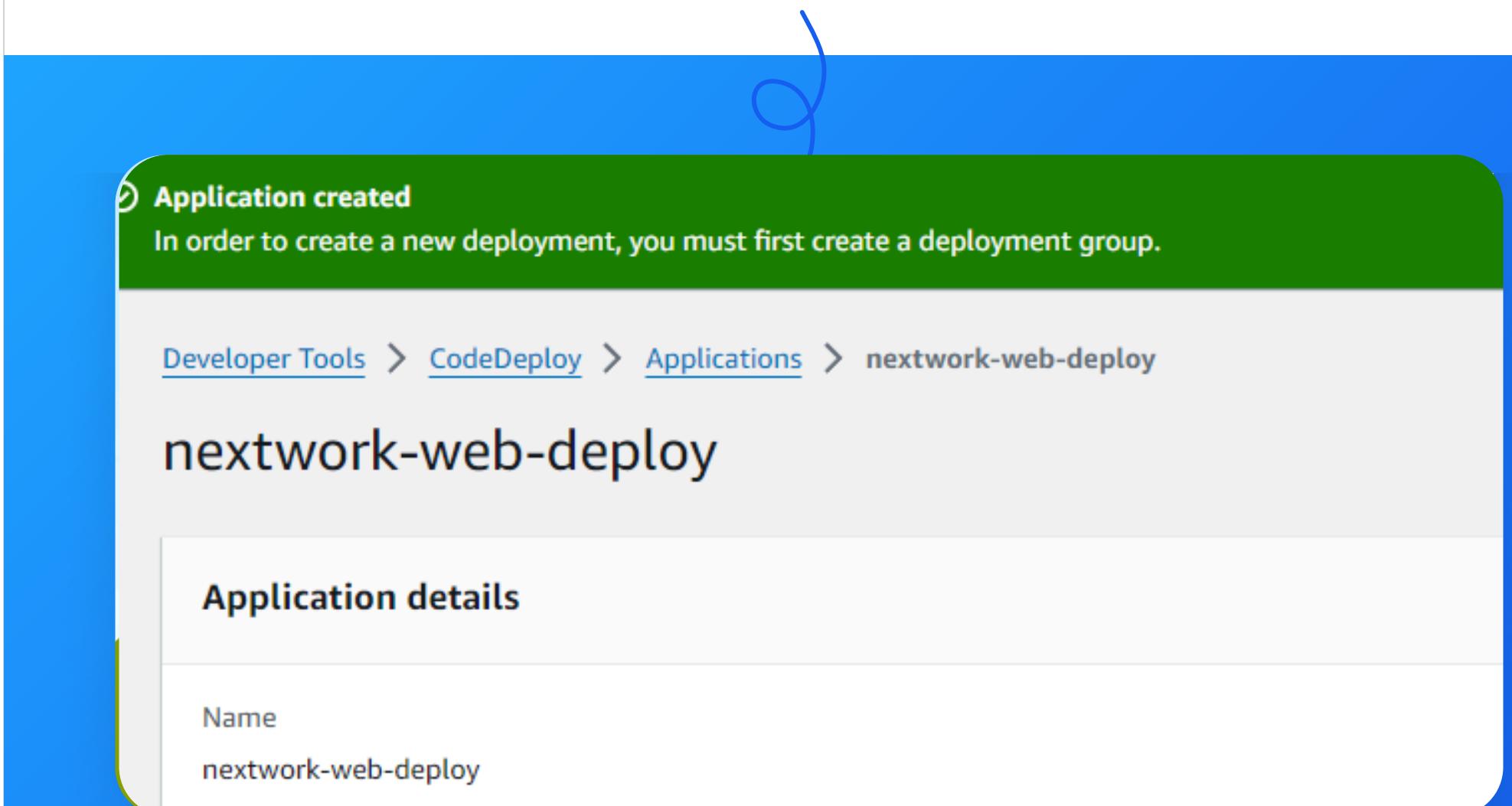
```
1 - [ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [  
7                 "autoscaling:CompleteLifecycleAction",  
8                 "autoscaling>DeleteLifecycleHook",  
9                 "autoscaling:DescribeAutoScalingGroups",  
10                "autoscaling:DescribeLifecycleHooks",  
11                "autoscaling:PutLifecycleHook",  
12                "autoscaling:RecordLifecycleActionHeartbeat",  
13                "autoscaling>CreateAutoScalingGroup",  
14                "autoscaling>CreateOrUpdateTags",  
15                "autoscaling:UpdateAutoScalingGroup",  
16            ]  
17        }  
18    ]  
19}
```



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 interface. At the top, there is a green banner with the text "Application created" and a note: "In order to create a new deployment, you must first create a deployment group." Below the banner, the navigation path is displayed: "Developer Tools > CodeDeploy > Applications > nextwork-web-deploy". The main title of the application is "nextwork-web-deploy". Underneath the title, there is a section titled "Application details" which shows the "Name" as "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.

Kanika Mathur
github.com/KanikaGenesis

NextWork.org

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.

arn:aws:iam::991380288324:role/NextWorkCodeDeployRole

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 CodeDeploy Agent.

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

14 Days ▾



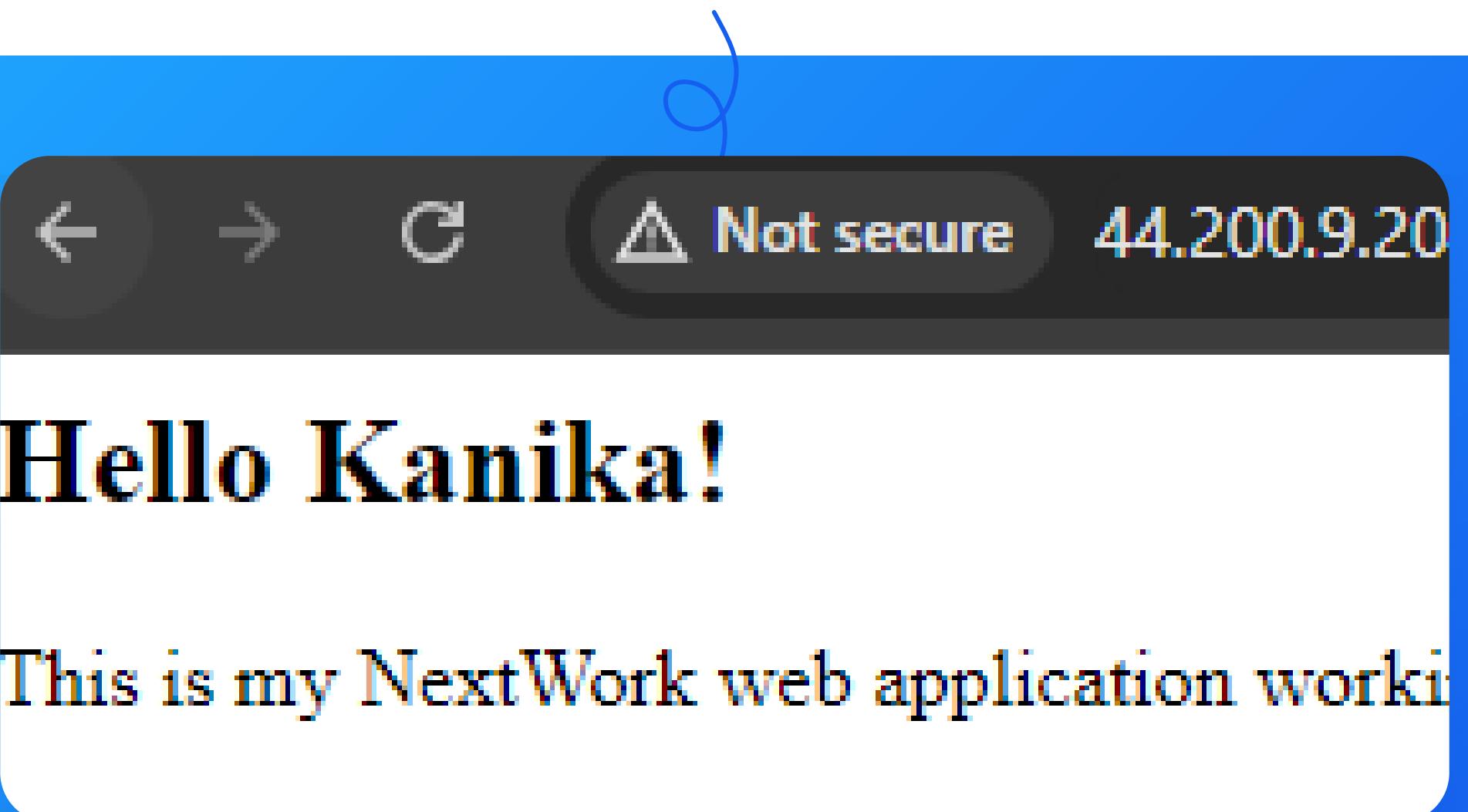
Kanika Mathur
github.com/KanikaGenesis

NextWork.org

Deployment success! 🚀

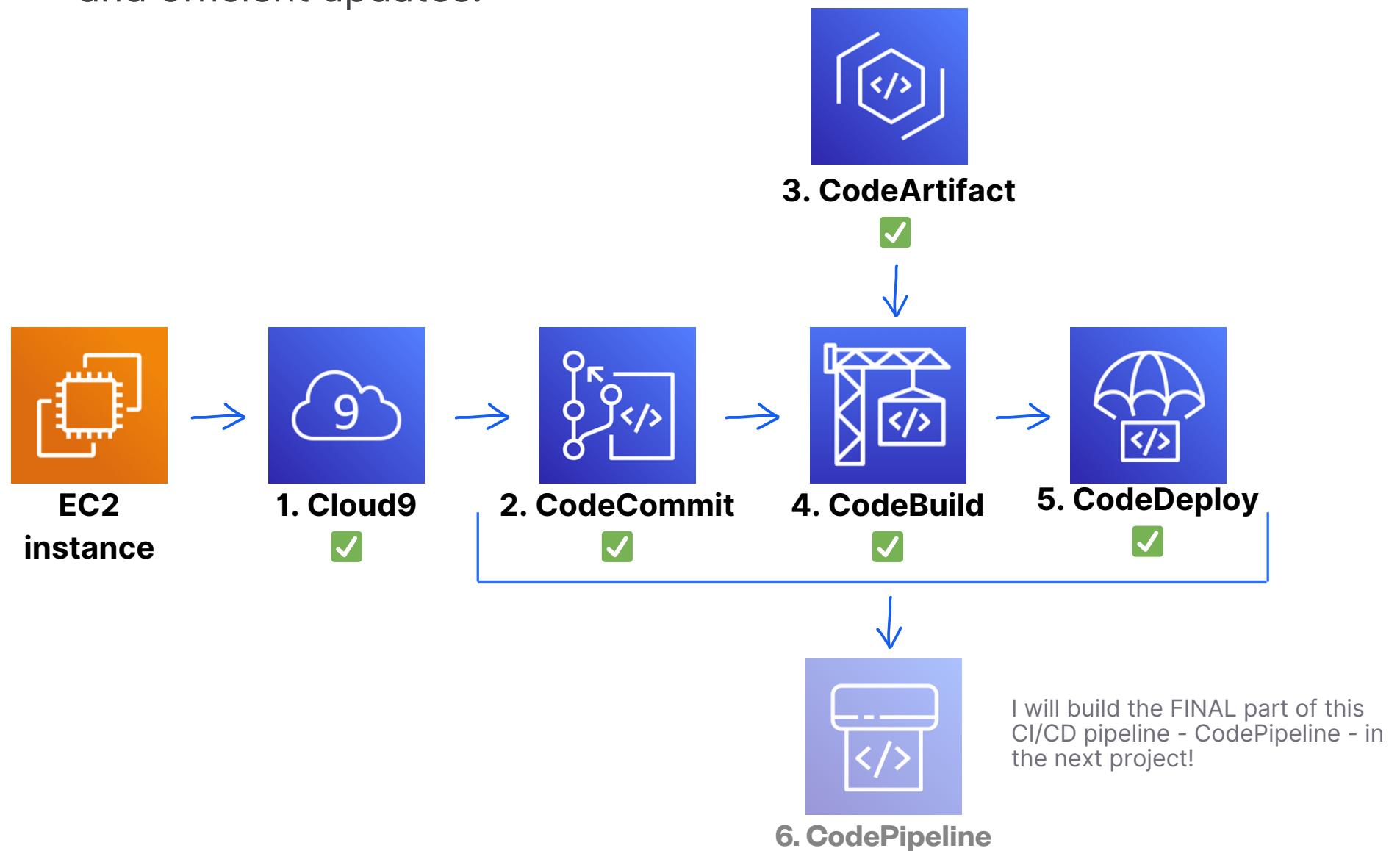
- After setting up my deployment group, I was ready to kick off my first deployment!
- To create my deployment, I had to set up a revision location, which means the location of my web app's build artifacts.
- My revision location was the Java WAR file/zip file containing my compiled web app code. It 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.





Set Up a CI/CD Pipeline

- A CI/CD pipeline is a practice of continuous integration, continuous deployment. This makes sure that changes made to source code is constantly being integrated/updated in a shared Git repository, so that other developers working from the same code base are always the latest version of the code (continuous integration). It also makes sure that end users are always seeing the latest version/updates to an application (continuous deployment).

To set up a CI/CD in CodePipeline, I configured three stages:

- The **source** stage, which means the source code for my web app. This is currently stored in CodeCommit.
- The **build** stage, which means the service that is managing the build process for my web app (i.e. compiling and packaging my web app into a handy WAR file.). This CodeBuild.
- The **deploy** stage, which means the service that is managing the deployment process for my web app (i.e. making my web app available to the world). This is CodeDeploy.

kanika Mathur
github.com/KanikaGenesis

NextWork.org

A peek into my pipeline set up in CodePipeline!

nextwork-web-pipeline

Pipeline type: V2 Execution mode: SUPERSEDED

⌚ Source Succeeded

Pipeline execution ID: [0657fecf-8c59-4025-8b4b-389d67edfc20](#)

Source

[AWS CodeCommit](#)

⌚ Succeeded - Just now

[50f4f606](#)

[View details](#)

[50f4f606](#) Source: Adding CodeDeploy files

[Disable transition](#)

⌚ Build ⓘ In progress

Pipeline execution ID: [0657fecf-8c59-4025-8b4b-389d67edfc20](#)



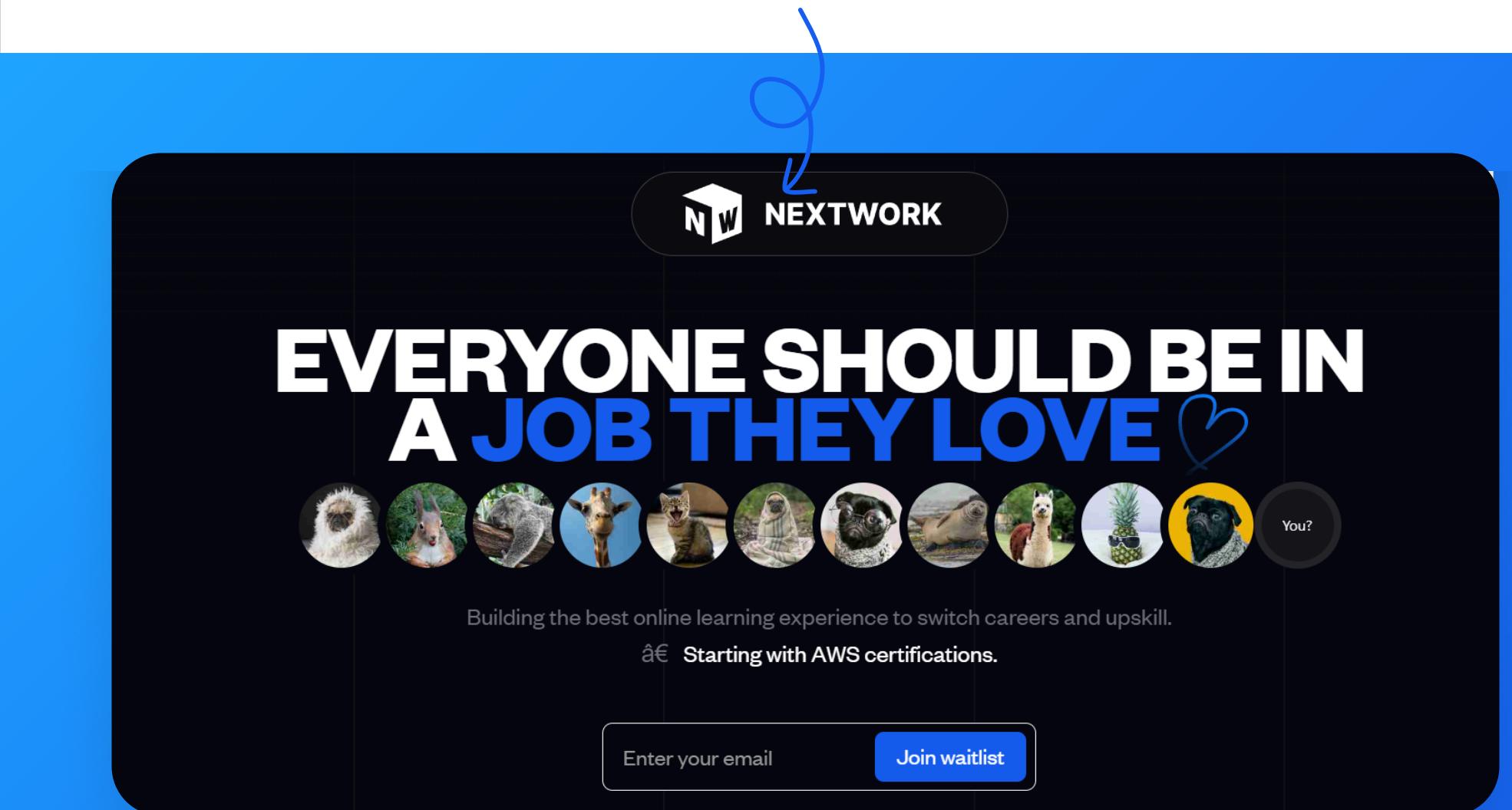
kanika Mathur
github.com/KanikaGenesis

NextWork.org

Release a Change

- My CI/CD pipeline gets triggered by a commit in my local working environment (Cloud9) - which updates my CodeCommit repository.
- I tested this by making **two** updates to my web app's source code. These changes were an edit to **index.jsp** and uploading a folder of image assets in the **webapp** folder of my project files.
- Once my pipeline executed successfully, I checked the IPv4 address of my web server that is hosting my web app.

My web app automatically updated after I committed a change.





Trigger A Rollback

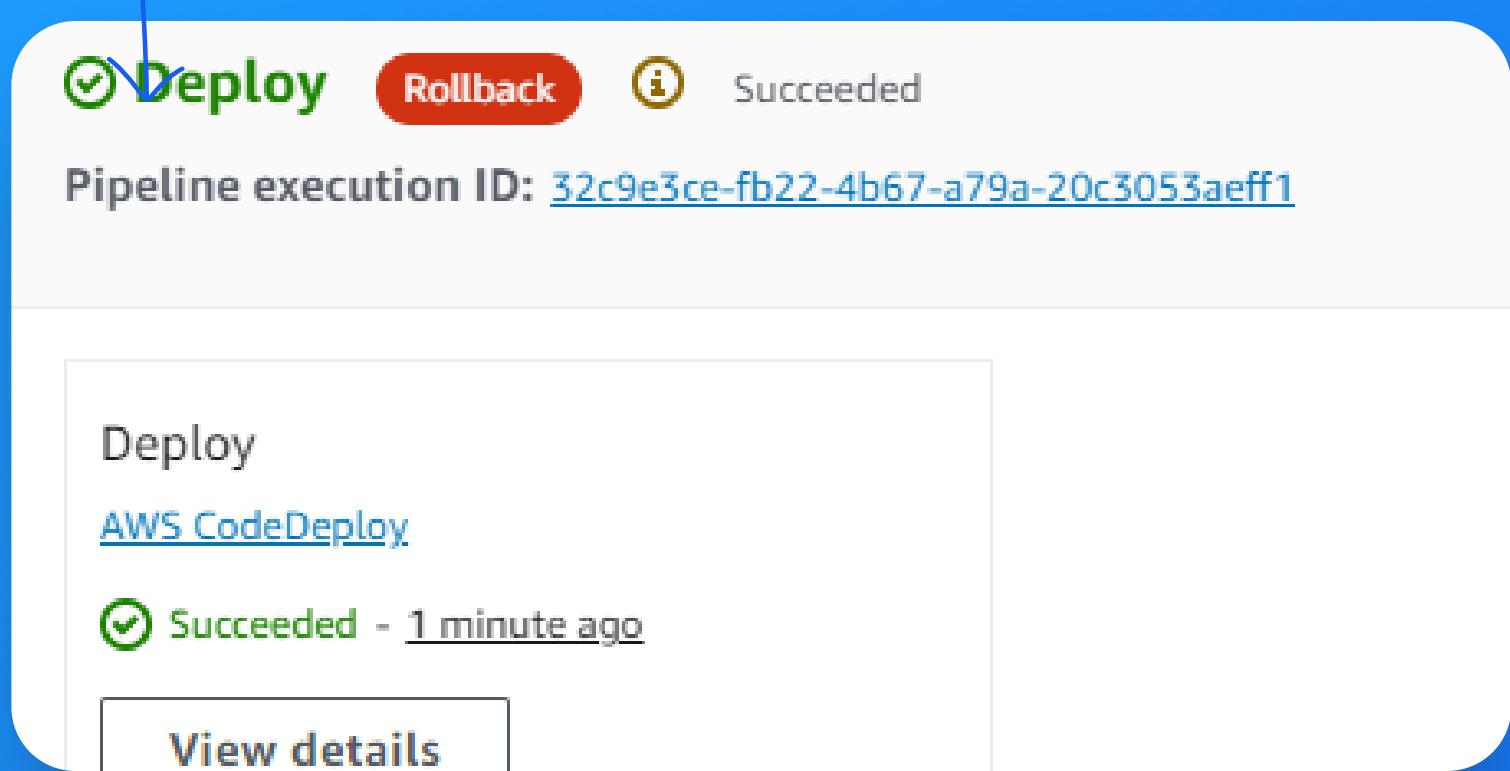
- A rollback in a pipeline means reverting the version of our code that a stage of our pipeline is referring to.
- I initiated a rollback on the deploy stage. I checked the source stage, and learnt that the source stage was unaffected by the rollback, and stayed using the latest version of my source code.
- Once the rollback was complete, my web app's appearance reverted back to reflect the simple version of my index.jsp file before I ever updated my index.jsp file and uploaded new image assets.
- To update my Deploy stage back to the latest version of my source code, I **released** a change in my CodePipeline pipeline. Releasing a change means the latest version of my source code gets referenced across the Source, Build AND Deploy stages of my pipeline.



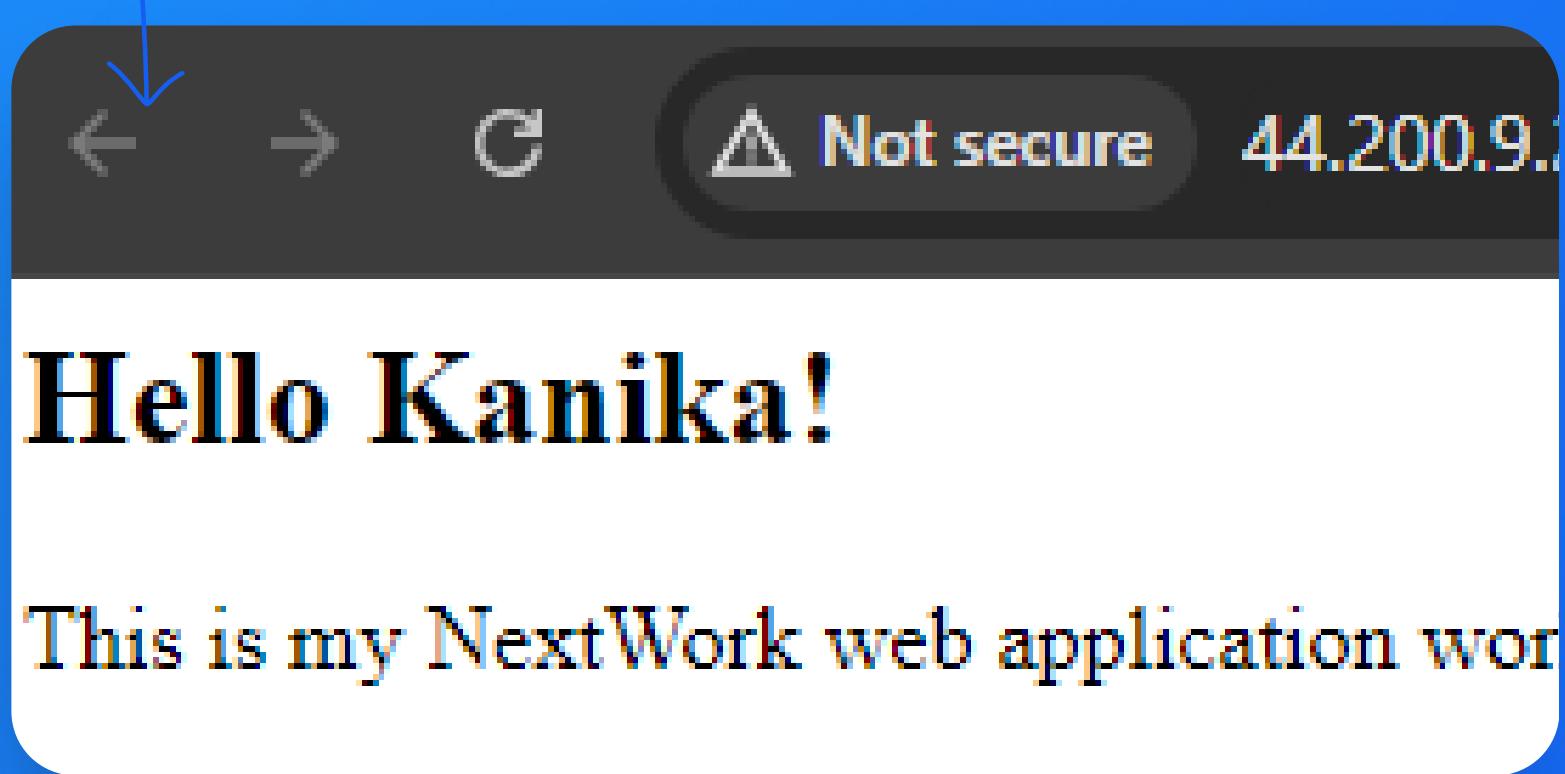
kanika Mathur
github.com/KanikaGenesis

[NextWork.org](#)

My CodeDeploy panel showing a successful rollback.

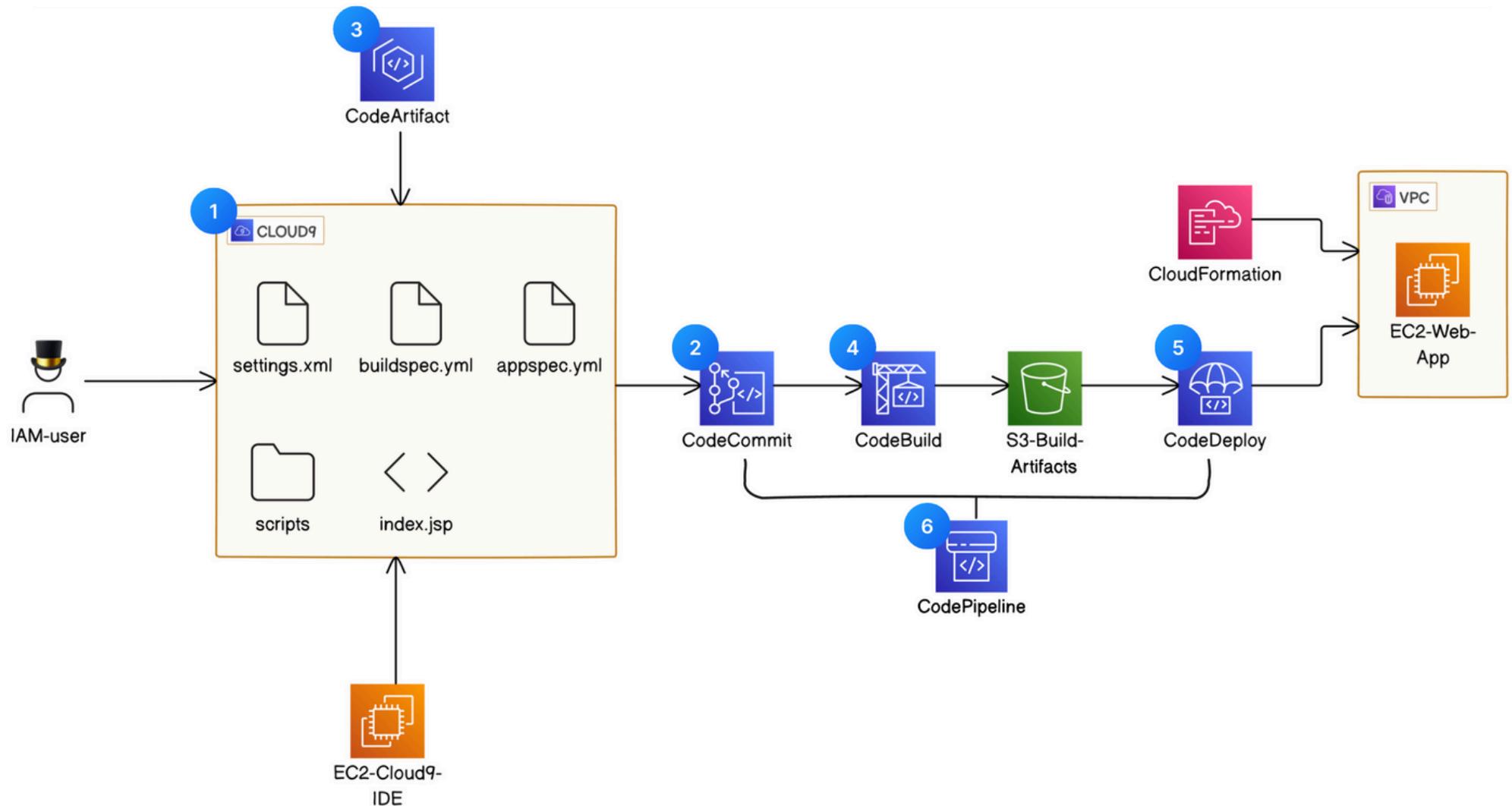


My web app returned to its previous state after the rollback.



My CI/CD Pipeline

1. **AWS Cloud9** is responsible for providing an integrated development environment (IDE) for writing, running, and debugging code in the cloud.
2. **AWS CodeCommit** is responsible for hosting secure Git repositories to store and version control the source code.
3. **AWS CodeArtifact** is responsible for managing software packages and dependencies, making it easy to store, publish, and share them.
4. **AWS CodeBuild** is responsible for compiling and packaging my app's source code into a handy WAR file.
5. **AWS CodeDeploy** is responsible for deploying that WAR file on servers, so users can see my web app!
6. **AWS CodePipeline** is responsible for automating the integrations between CodeCommit, CodeBuild and CodeDeploy.





My key learnings

1

Through this Cloud DevOps project, I gained a solid understanding of core principles such as **continuous integration**, **continuous delivery/deployment**, and **infrastructure as code**. Proficiency in using **AWS Cloud9** as an integrated development environment was a crucial skill acquired, enabling effective coding and debugging.

2

Mastering **Infrastructure as Code (IaC)** using **AWS CloudFormation** allowed me to define and provision infrastructure resources efficiently. This approach emphasized the benefits of version-controlled infrastructure and repeatable deployments, ensuring consistency and reliability. Implementing a **CI/CD** pipeline with **AWS CodePipeline**, **CodeBuild**, and **CodeDeploy** automated the build, test, and deployment processes, significantly improving the release cycle's speed and reliability.

3

Integrating various AWS services such as **S3**, **EC2**, and **IAM** into the DevOps workflow provided hands-on experience in configuring permissions and managing access control. This integration allowed for seamless data storage, compute resource management, and secure user authentication.

4

Throughout the project, I faced and overcame challenges related to environment configuration, service integrations, and pipeline automation. These experiences enhanced my problem-solving skills and underscored the importance of best practices like automated testing, incremental deployments, and continuous feedback loops. The project has prepared me to apply the knowledge and skills acquired to real-world scenarios and future projects.



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

Check out community.nextwork.org for more free projects

