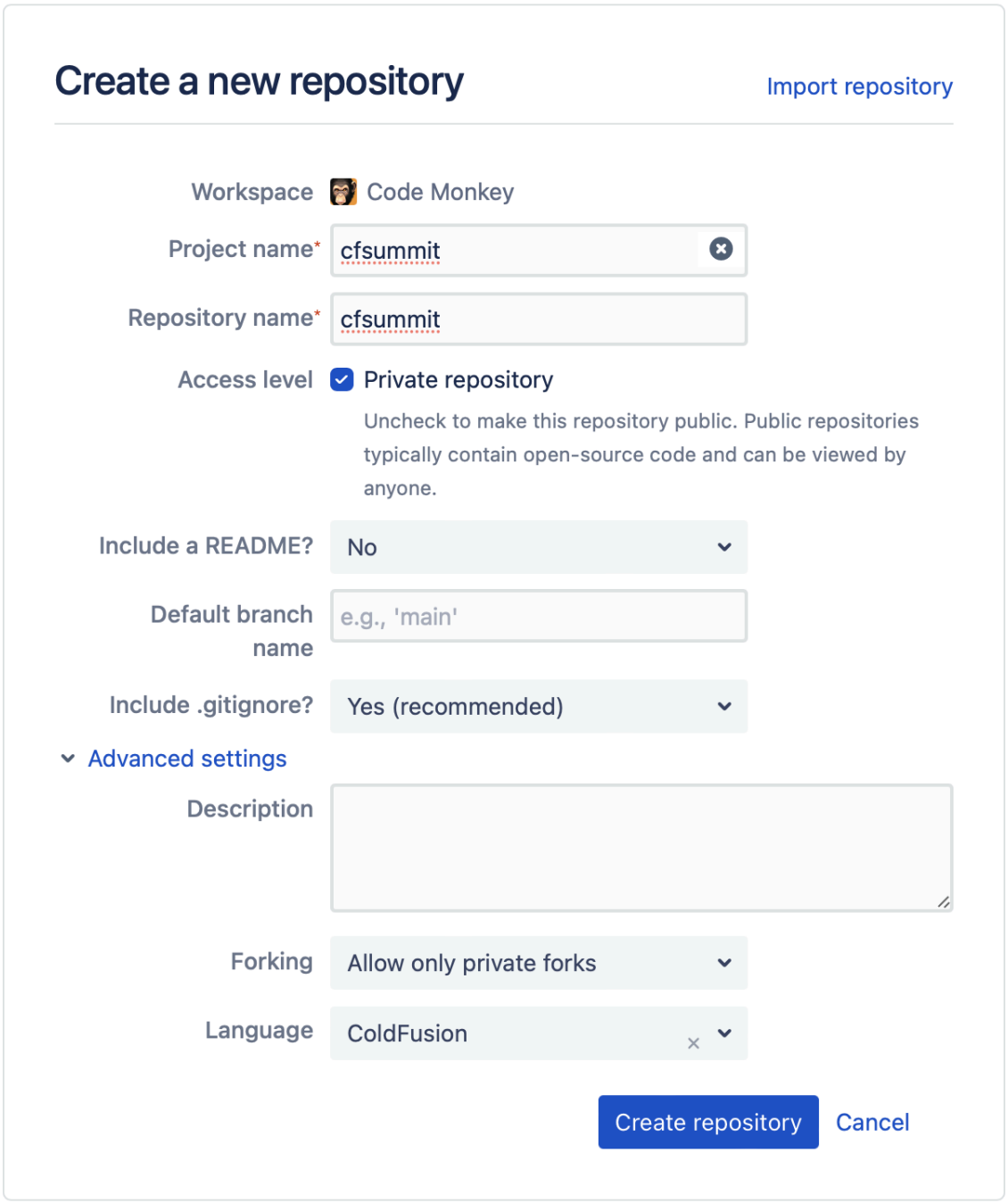
# Part 1 - Create a Local Development Environment

## Step 1 - Establish the Repository

- ☐ Log into BitBucket and create a new repository
  - ☐ Create a new project
  - ☐ Create a new repository
  - ☐ Assign a default branch name, such as production
  - ☐ Select ColdFusion as the language




- [ ] Clone the repository into your local project folder.

## Step 2 - Local Repository Setup

- [ ] In the project directory, create an `app` directory.
- [ ] In the `app` directory, create an `index.cfm` file.
- [ ] In the `index.cfm` file, paste the following code:

```
<h2>cfsummit.vegas</h2>
<cfoutput>
```

```
    <p>#server.coldfusion.productName# (#server.coldfusion.productLevel#)
version #server.coldfusion.productVersion#</p>
    <p>#dateTimeFormat( now(), 'long' )#</p>
</cfoutput>
```

## Step 3 - Use the CF Docker Image to create a local development environment

### On an Intel / AMD64 / Windows environment

- ☐ In VS Code, create a file called `docker-compose.yml` in the root of the project folder with the following content:

```
version: "3.8"

services:
  cfsummit:
    image: adobecoldfusion/coldfusion:latest
    platform: linux/amd64
    container_name: cfsummit
    hostname: cfsummit
    environment:
      - acceptEULA=YES
      - password=cfsummit
      - enableSecureProfile=false
      - installModules=all
    ports:
      - 8500:8500
    volumes:
      - ./app:/app

networks:
  cfsummit_network:
```

### On an ARM based (Apple Silicon) environment

Adobe has not yet created Docker images that work seamlessly on Apple Silicon.  Due to this, I can't demonstrate a Docker environment using the ColdFusion Docker images on a Mac. Instead, I will use the Ortus Solutions CommandBox Docker images to spin up ColdFusion 2023.

- [ ] In Visual Studio Code, create a file called `docker-compose.yml` in the root of the project folder with the following content:

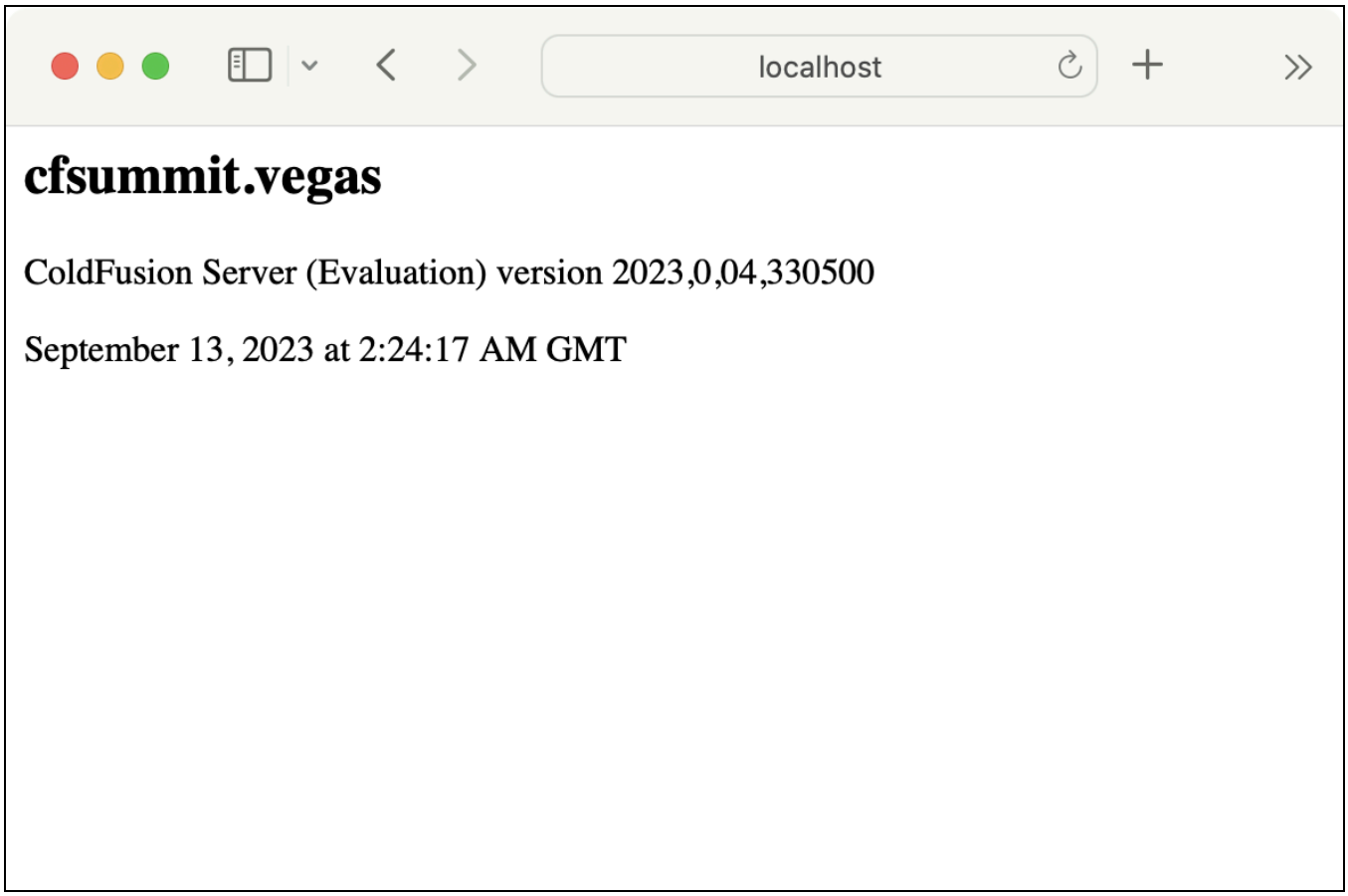
```
version: "3.8"

services:
  cfsummit:
    image: ortussolutions/commandbox:adobe2023
    container_name: cfsummit
    hostname: cfsummit
    environment:
      - BOX_INSTALL=true
      - cfconfig_adminPassword=cfsummit
      - TZ="US/Pacific"
    ports:
      - 80:8080
      - 8500:8500
    volumes:
      - ./app:/app

networks:
  cfsummit_network:
```

Once the `docker-compose.yml` file is created

- [ ] In a command line terminal, browse to the project folder and enter `docker compose up -d`.
- [ ] Check to see that the server is up and running from the Docker Dashboard.
- [ ] In a browser, browse to `localhost`. The rendered contents of our `index.cfm` file should resemble the screenshot below. Refreshing the page will update the timestamp.




Let's make that look better

- [ ] Replace the contents of the `index.cfm` file with the following:

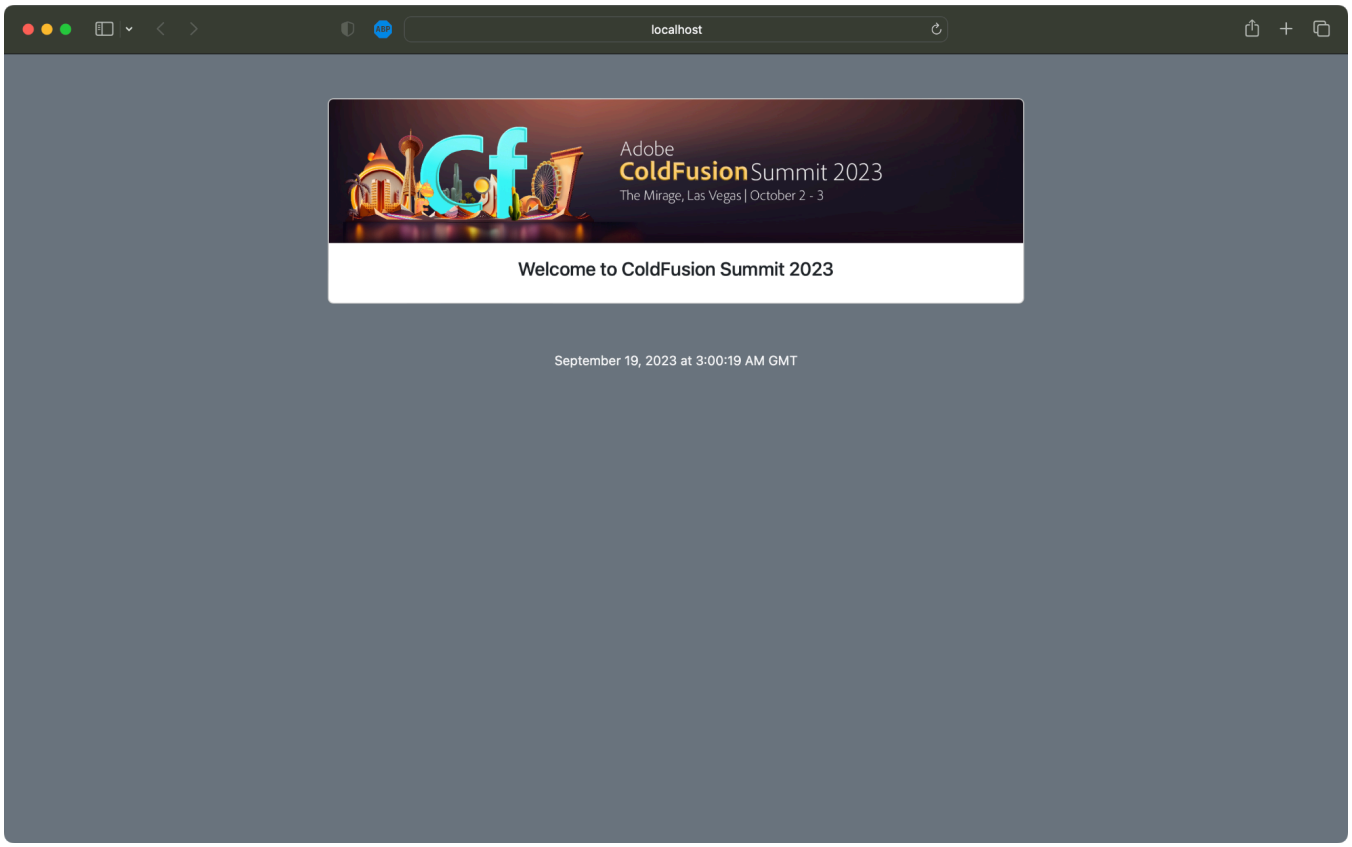
```
<!doctype html>
<html lang="en">
      <head>
            <meta charset="utf-8">
            <meta name="viewport" content="width=device-width, initial-scale=1">
            <title>cfsummit.vegas</title>

            <!--- Bootstrap 5.3.1 --->
            <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">
            <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.bundle.min.js"
integrity="sha384-HwwvtgBNo3bZJJLYd8oVXjrBZt8cqVSpeBNS5n7C8IVInixGAoxmnlMuBnhbgrkm" crossorigin="anonymous"></script>
```

```
    </head>

    <body class="bg-dark py-5">
        <cfoutput>
        <div class="container">
            <div class="row">
                <div class="col-11 col-sm-10 col-md-9 col-lg-8 col-xxl-7
mx-auto">
                    <div class="card">
                        <img
src="https://block-image-uploader-prod.s3.us-west-2.amazonaws.com/901a6faf-d895-
43d1-a424-a17ecc688a46/CF_Summit_2023-desktop.jpg" class="card-img-top"
alt="ColdFusion Summit 2023">
                        <div class="card-body">
                            <h5 class="card-title text-center">Welcome
to ColdFusion Summit 2023</h5>
                        </div>
                    </div>
                    <p class="text-light text-center
my-5"><small>#dateTimeFormat( now(), 'long' )#</small></p>
                </div>
            </div>
        </div>
        </cfoutput>
    </body>
</html>
```

- [ ] Refresh the browser. The rendered contents of our `index.cfm` file should resemble the screenshot below. Refreshing the page will update the timestamp.




# Part 2 - AWS Identity Access Management (IAM) Setup

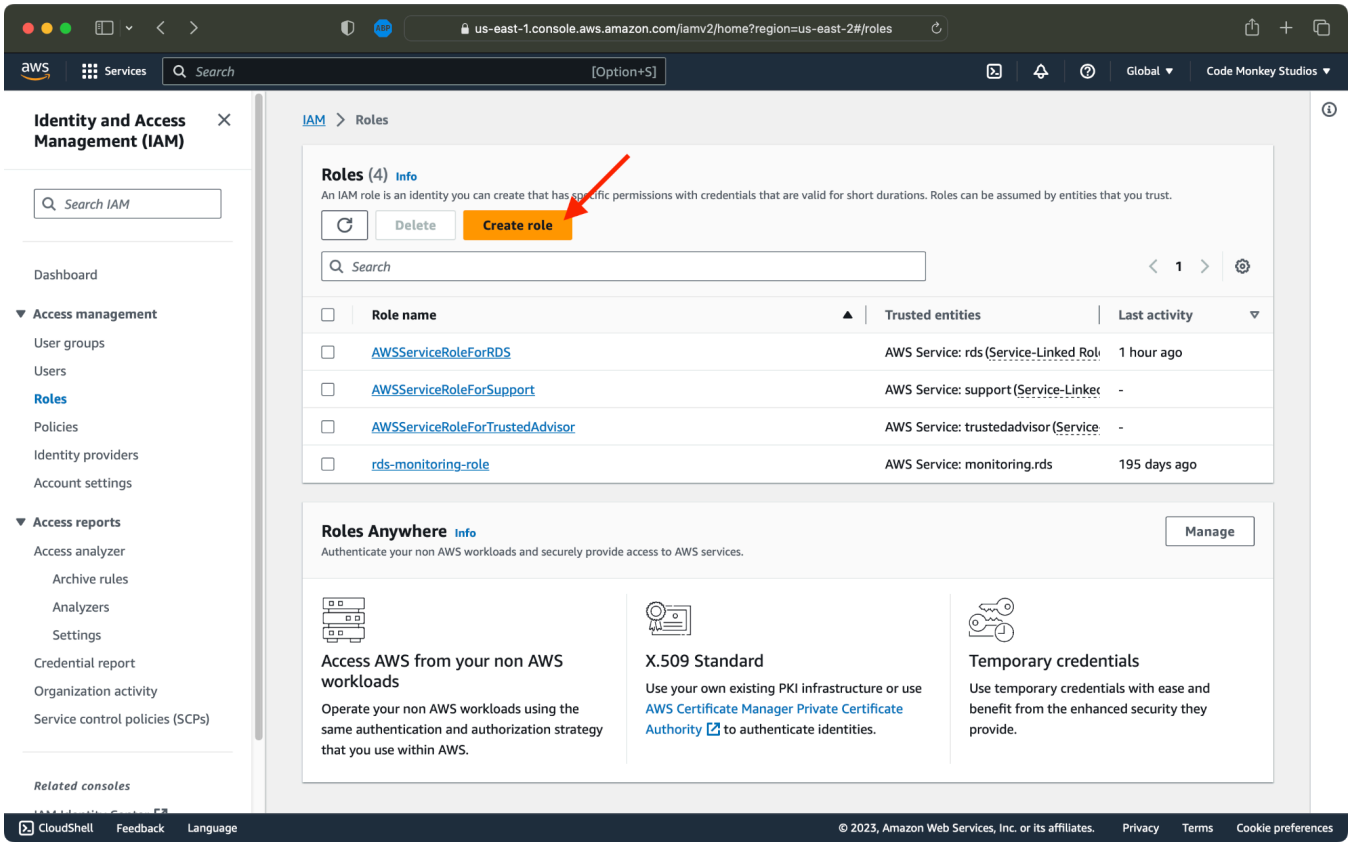
During this part, we are going to set up IAM on AWS for the user that will be used to push code via BitBucket Pipelines and AWS CodeDeploy .  We will create three elements in AWS IAM:

- A Role,
- A User,
- and A Group

## IAM Role

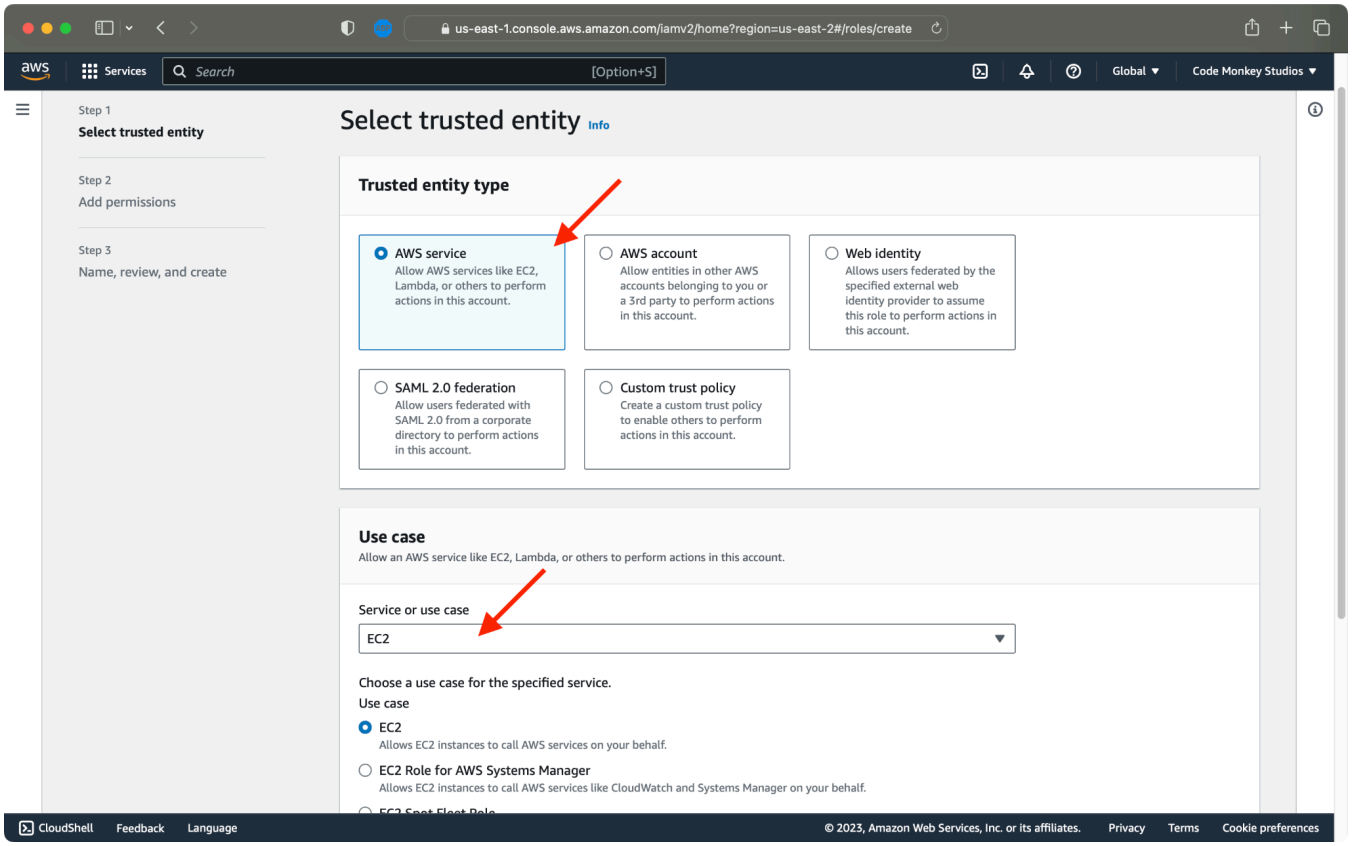
Let's start with the IAM Role, since the user will use this role to be granted permissions to push the code using AWS CodeDeploy.

- ☐ From the IAM Dashboard, select "Roles" in the left column.
- ☐ Click on the "Create Role" button.



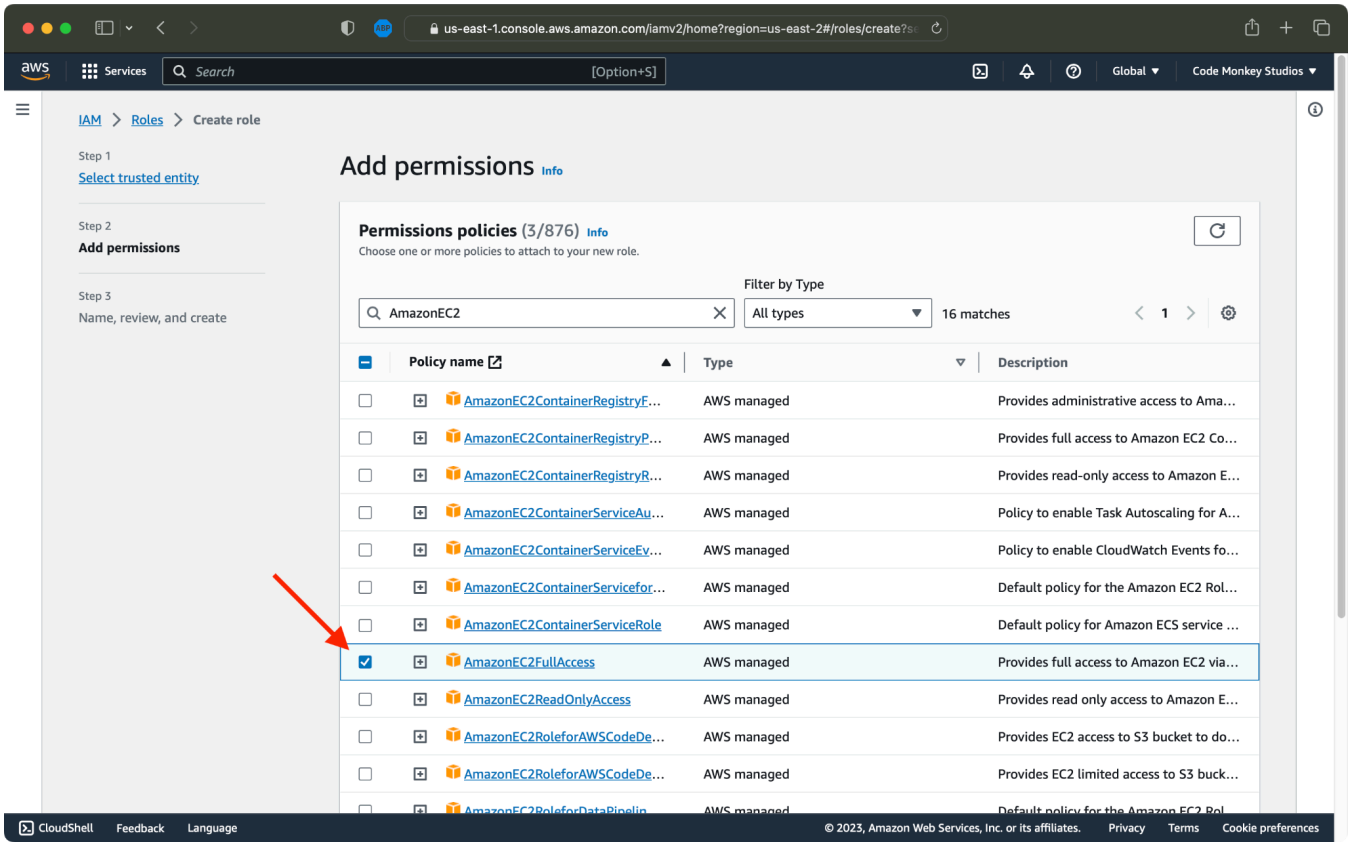

## Step 1 - Select trusted Entity

- ☐ Select “AWS Service” as the Trusted Entity Type.
- ☐ Select “EC2” as the Use Case.
- ☐ Click the “Next” button at the bottom of the page.



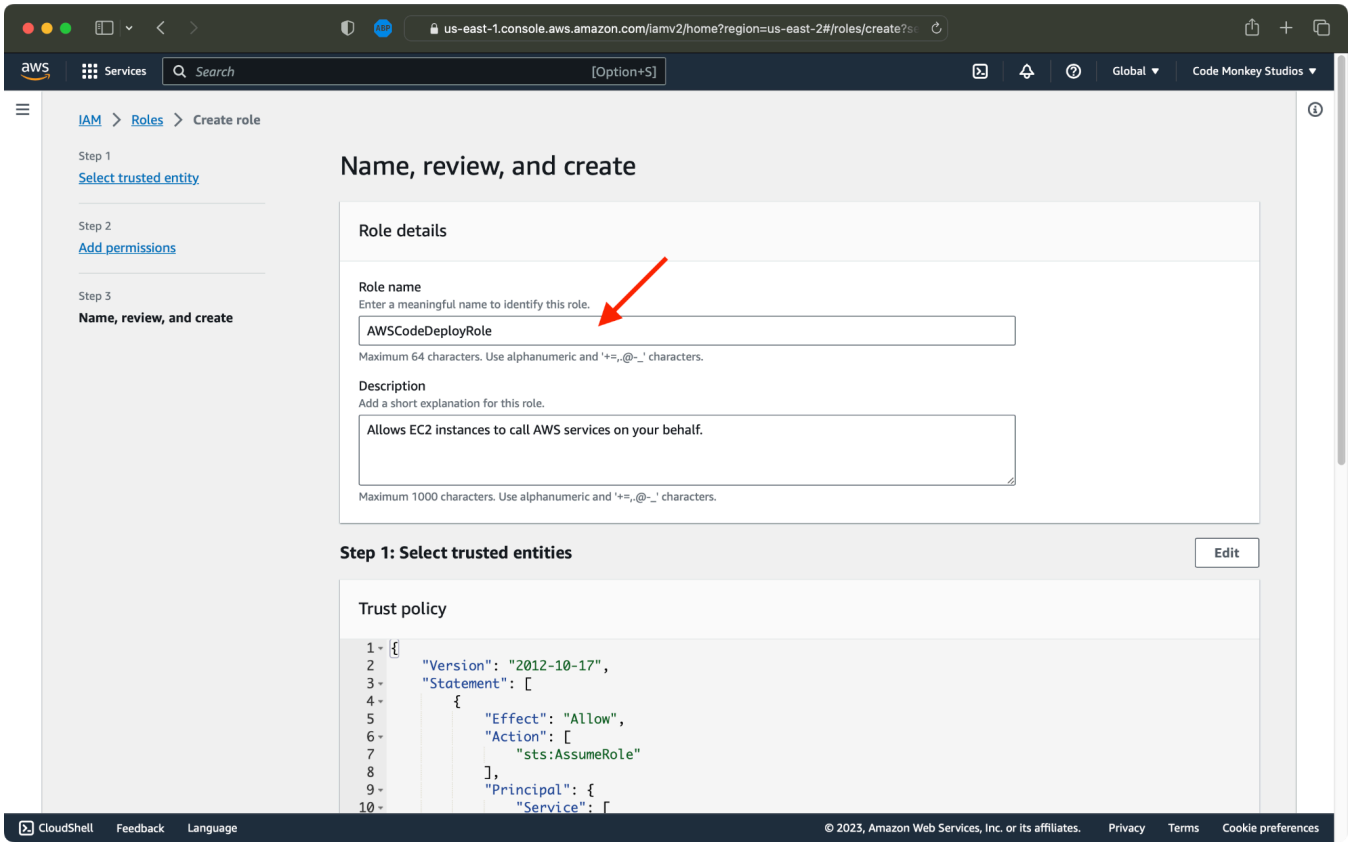

## Step 2 - Add Permissions

- ☐ Add the following permissions to the Role:
  - ☐ AmazonEC2FullAccess
  - ☐ AmazonS3FullAccess
  - ☐ AWSCodeDeployFullAccess
- ☐ Click the “Next” button at the bottom of the page.



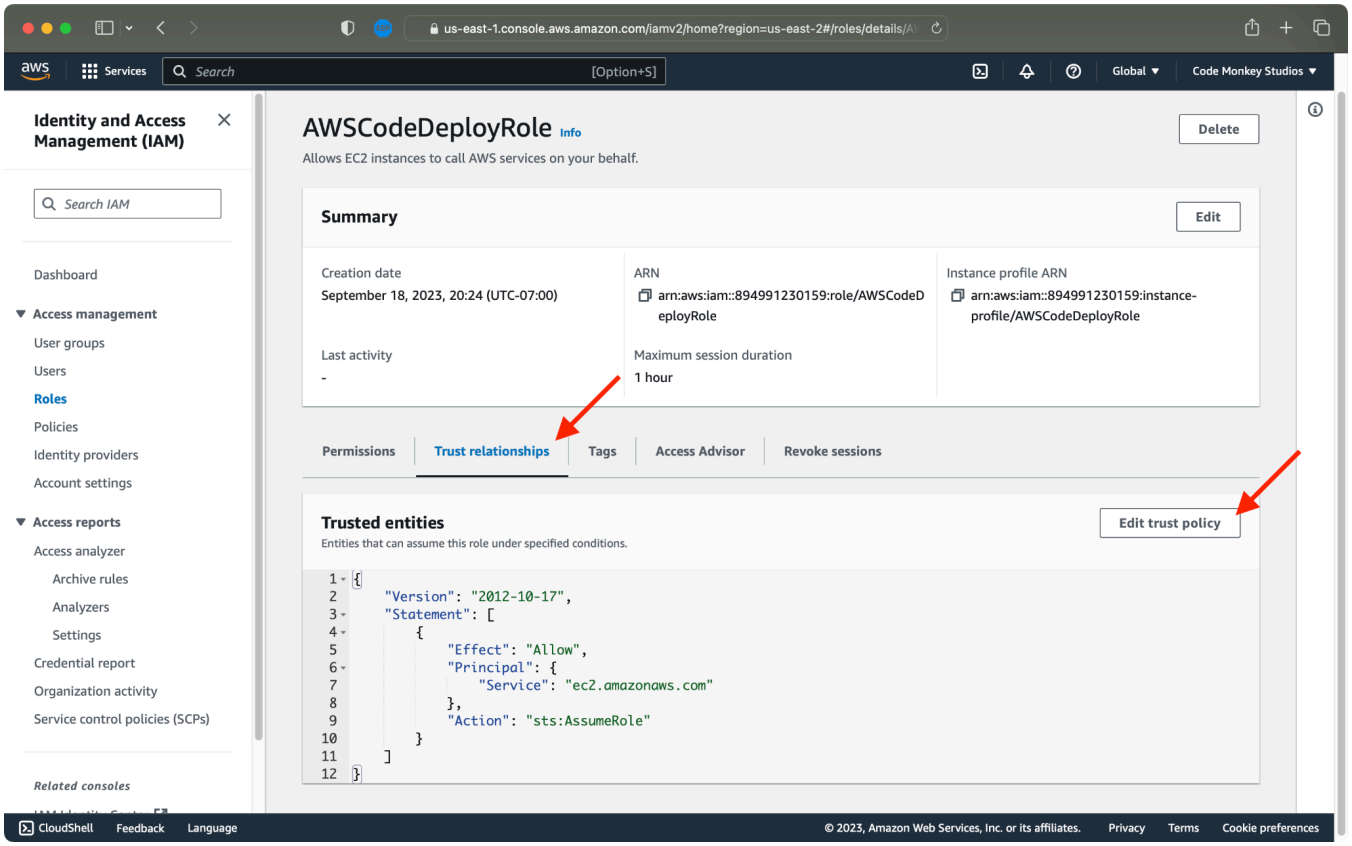

## Step 3 - Name, Review, and Create

- ☐ Name the Role `AWSCodeDeployRole`.
- ☐ Click the Create Role button at the bottom of the page.

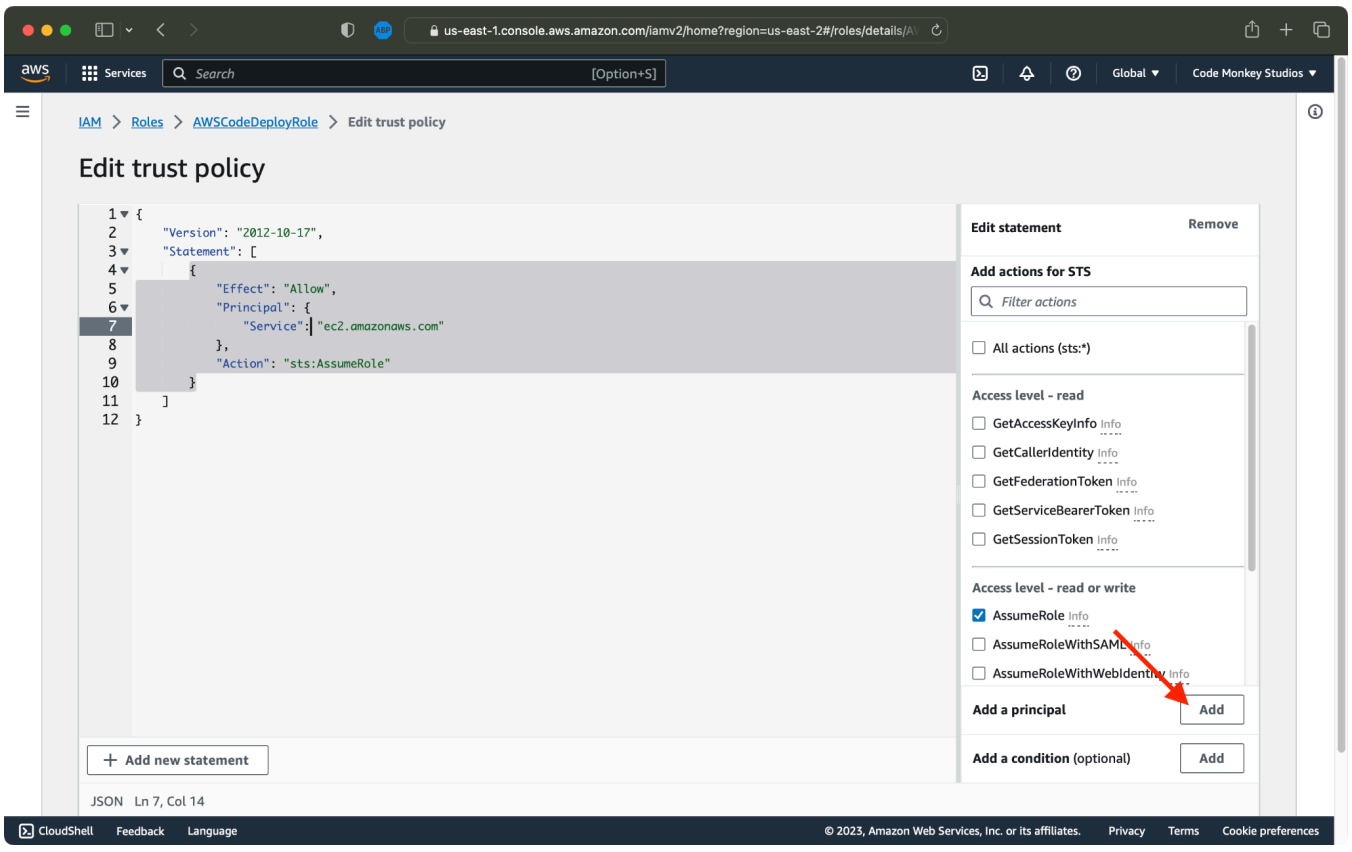



## Step 4 - Add Additional Principals to the Trust Policy

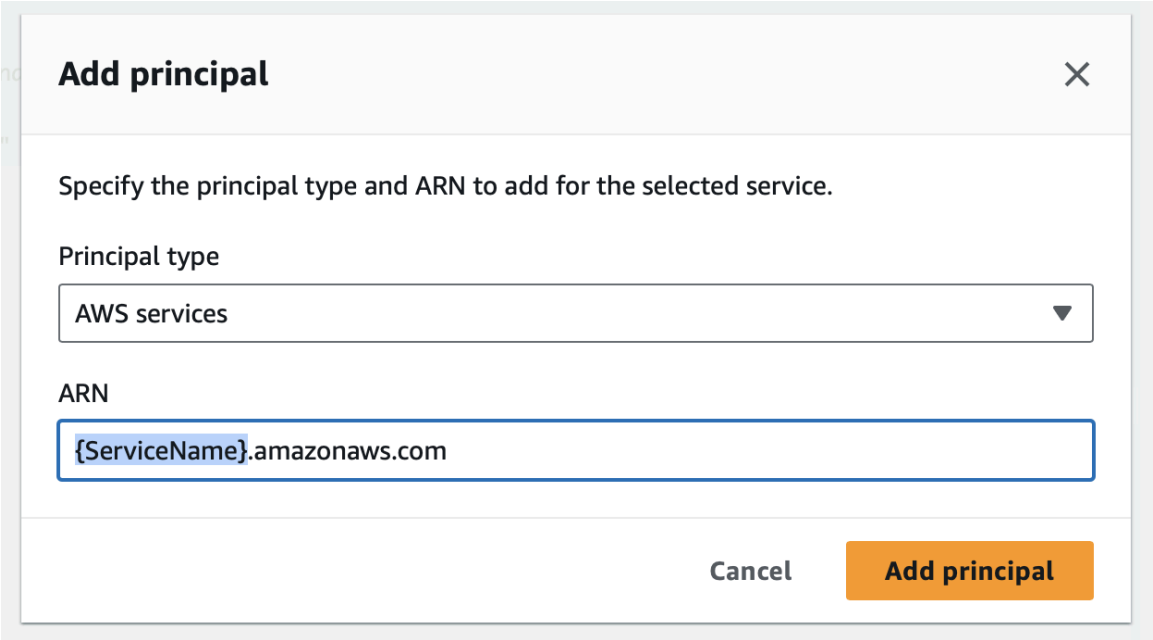
- ☐ From the IAM > Roles page, click on the newly created AWSCodeDeployRole role to edit it.
- ☐ Select the Trust Relationships tab.
- ☐ Click the Edit Trust Policy button.




☐ Click the Add button next to Add a Principal



- [ ] Under Principal Type select AWS Services.
- [ ] Under ARN, replace {ServiceName} with “codedeploy.YOUR_EC2_INSTANCE_REGION”. For example, if you intend to spin up your EC2 instances in the AWS region `us-east-2`, you would replace {ServiceName} with `codedeploy.us-east-2`.
- [ ] Click the Add principal button at the bottom of the modal window.




Once you add the CodeDeploy principal to the role, your trust policy should look like this:

```
{
     "Version": "2012-10-17",
     "Statement": [
          {
               "Effect": "Allow",
               "Principal": {
                    "Service": [
                         "ec2.amazonaws.com",
                         "codedeploy.us-east-2.amazonaws.com"
                    ]
               },
               "Action": "sts:AssumeRole"
          }
     ]
}
```

As a shortcut, you can copy and paste this JSON snippet directly into the Edit Trust Policy page.

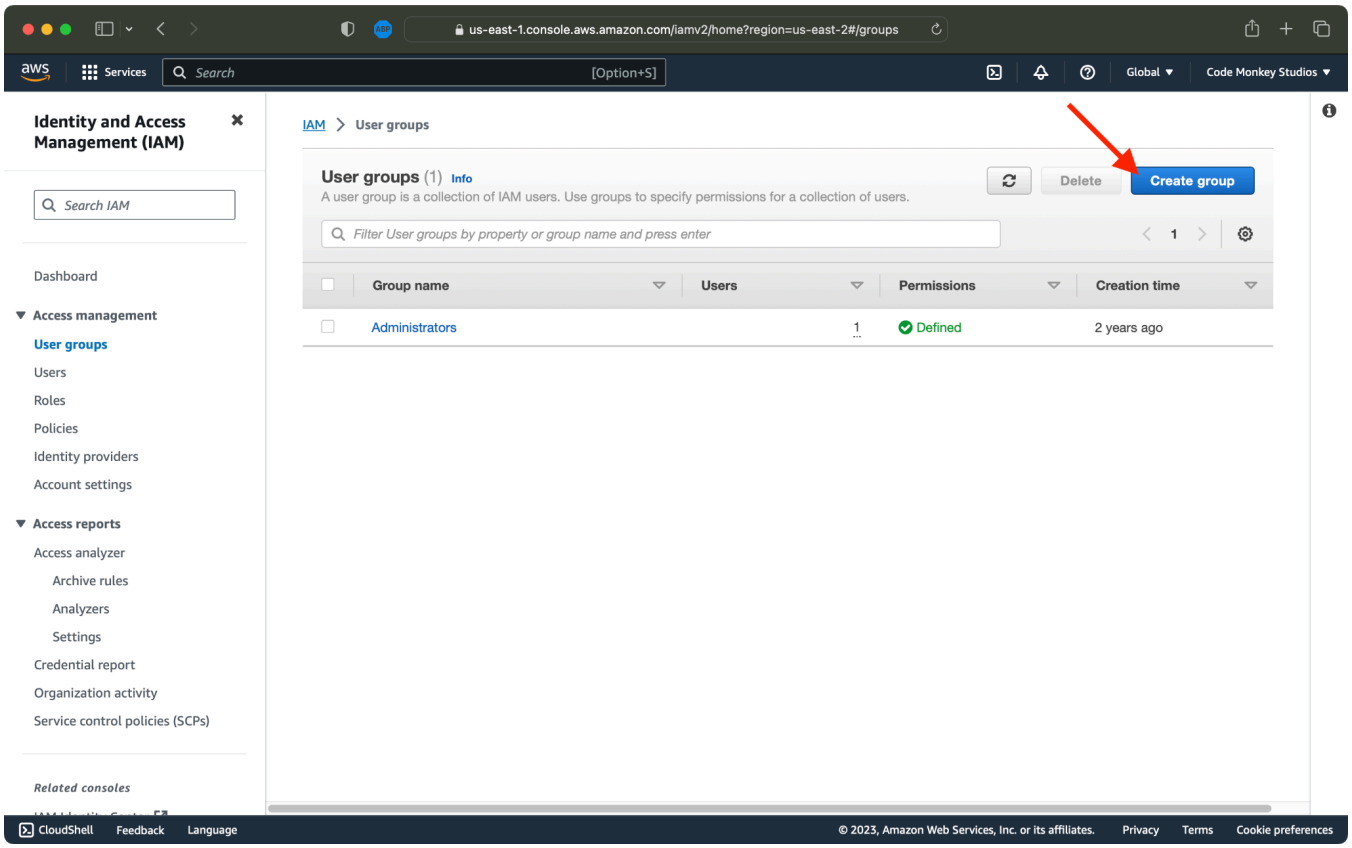
- [ ] Finally, click the “Update Policy” button at the bottom of the page.

The AWS IAM Role is now set up to be used by the IAM User, which we will create next.

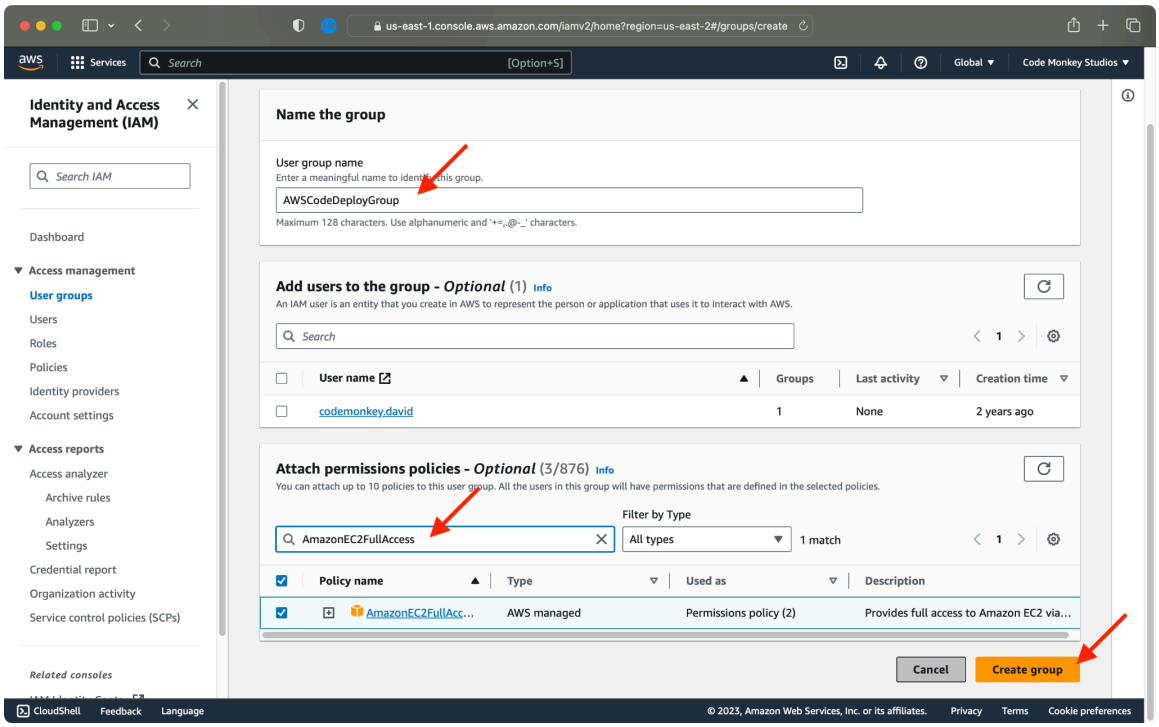
## IAM Group

Next, we will create an IAM Group for the User to join.  This group sets the permissions for the user to deploy code.

- [ ] From the IAM Dashboard, select “User Groups” in the left column.
- [ ] Click on the “Create Group” button in the upper right side of the user groups page.



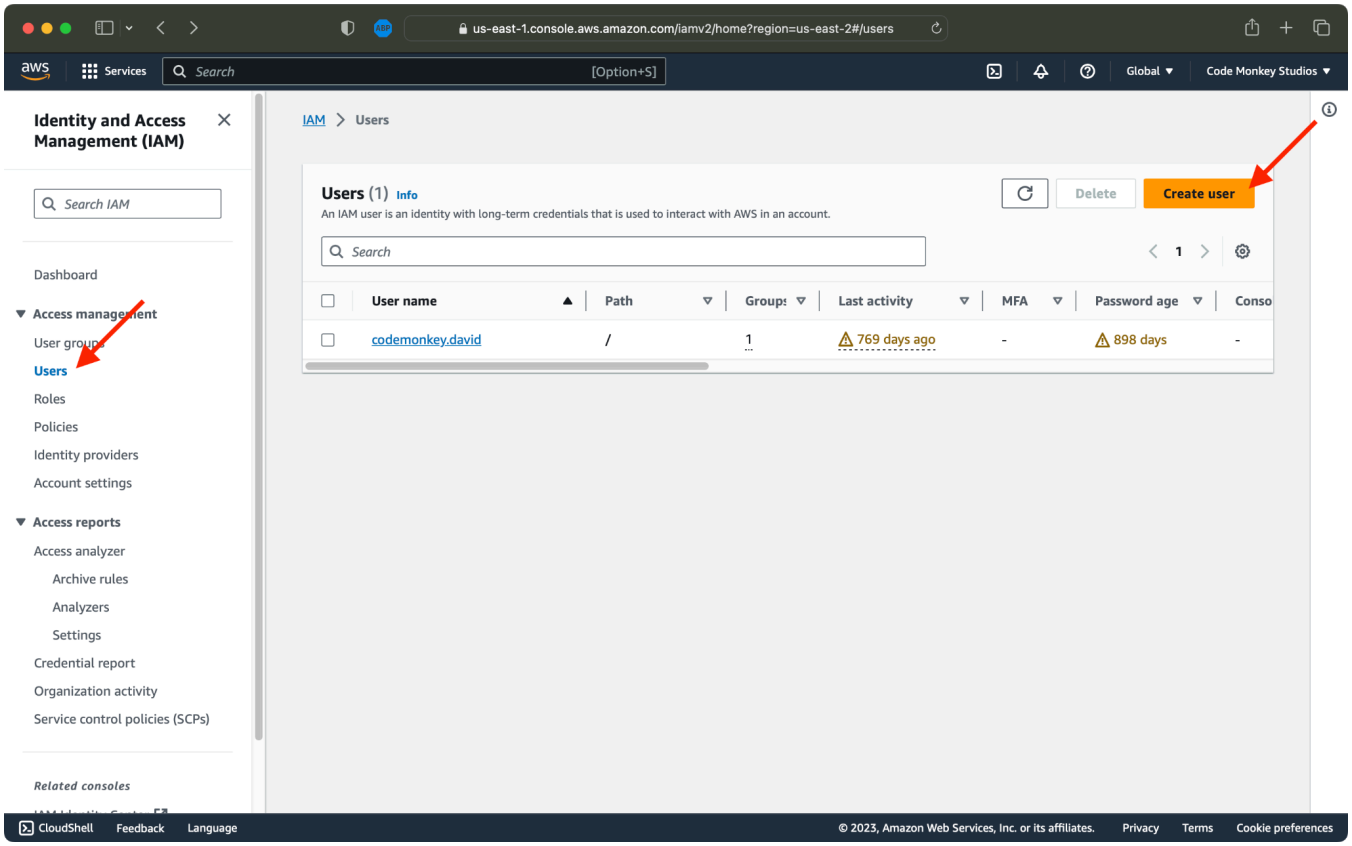

- [ ] Name the group AWSCodeDeployGroup.
- [ ] Add the following permissions to the Role:
    - [ ] AmazonEC2FullAccess
    - [ ] AmazonS3FullAccess
    - [ ] AWSCodeDeployFullAccess
- [ ] Click the “Create Group” button at the bottom of the page.




# IAM User

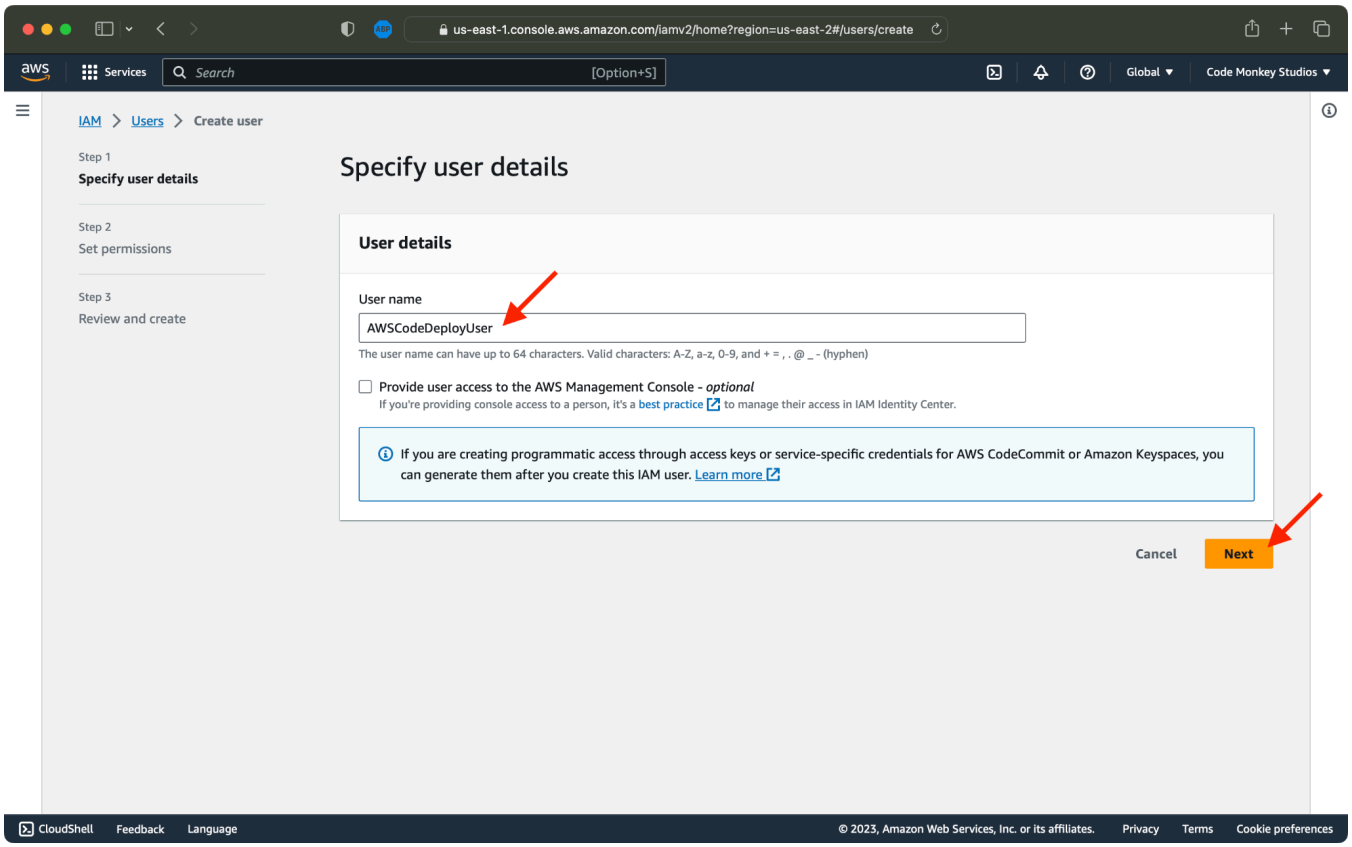
Finally, we will create an IAM User that will utilize the IAM Role and IAM Group we've created to deploy code using AWS CodeDeploy.

- ☐ From the left column of the AWS IAM Service, click on "Users" in the left column.
- ☐ Click on the "Create User" button on the right side of your screen.



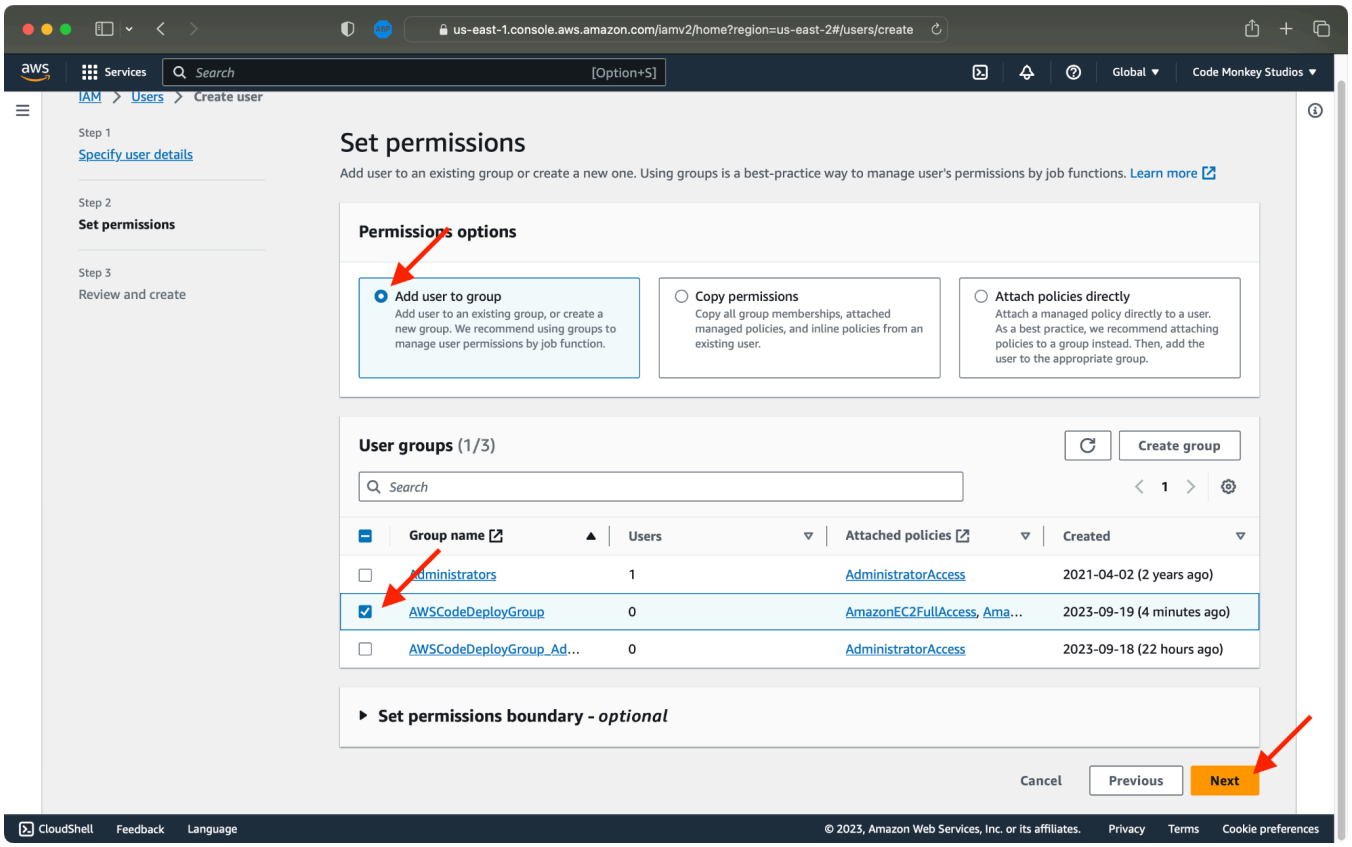

## Step 1 - Specify User Details

- ☐ Enter the username AWSCodeDeployUser.
- ☐ Click on the “Next” button at the bottom of the page.



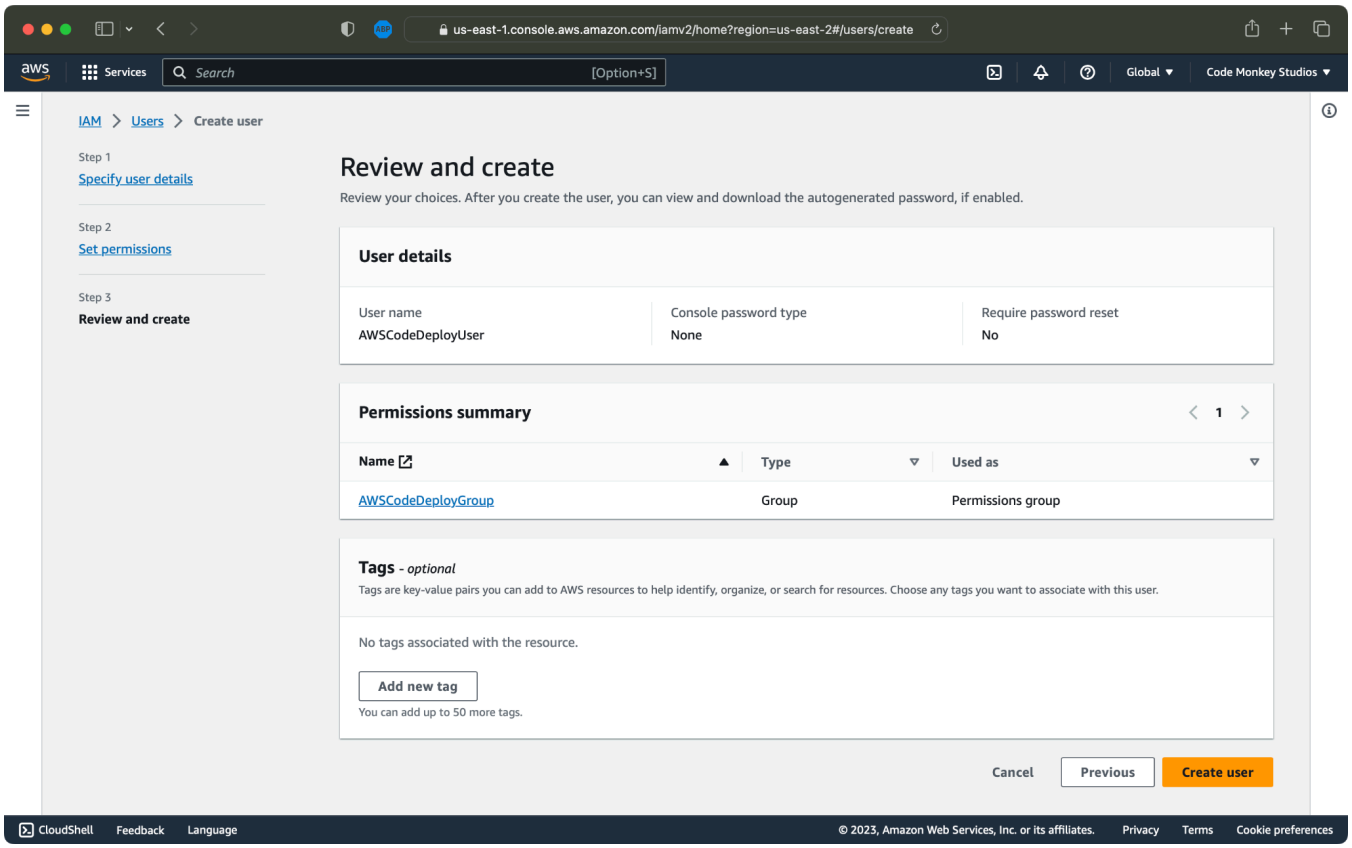

## Step 2 - Set Permissions

- ☐ Under Permissions options, select Add user to group.
- ☐ Under User groups, select the AwsCodeDeployGroup we created in the previous step.
- ☐ Click the Next button at the bottom of the page.




## Step 3 - Review and Create

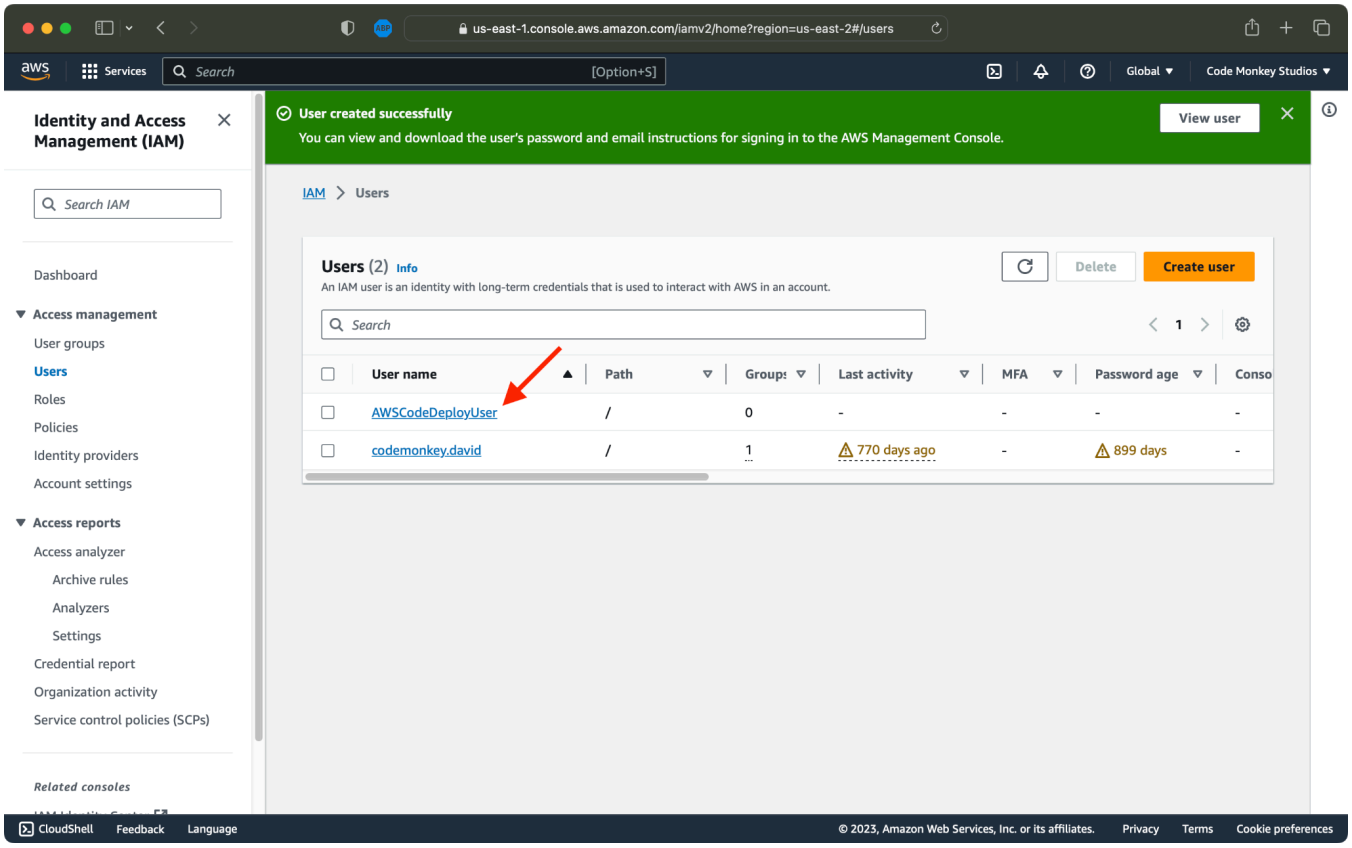
- [ ] On the Review and Create page, click the Create User button at the bottom of the page.



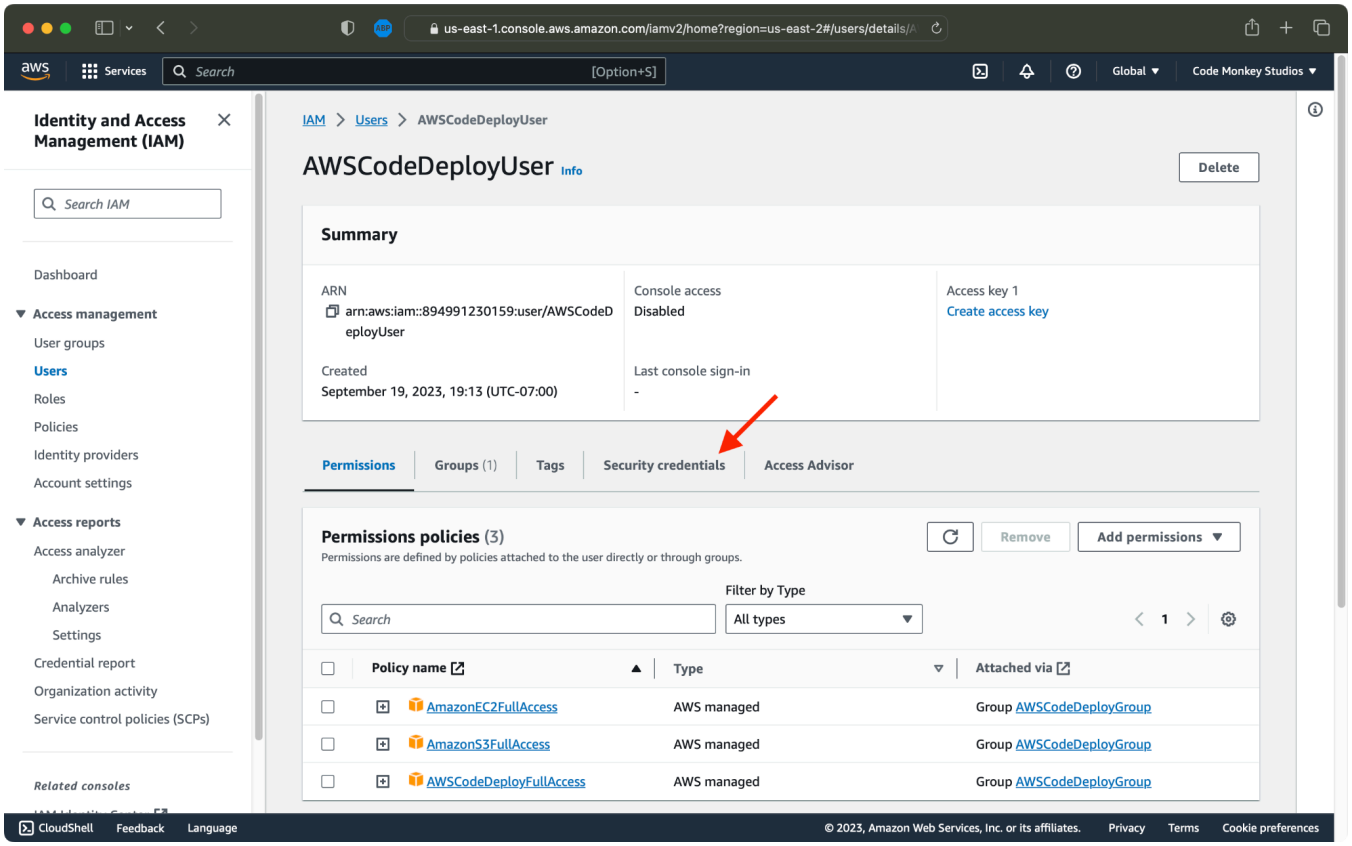
## Step 4 - Set Programmatic Access for the User

AWS takes security seriously, and they really don't want you creating long-lived access keys that live in perpetuity, so they're going to try and discourage you from doing what I'm suggesting. Like I stated at the top of my presentation, we are throwing security out the window in order to complete this task in under an hour. Use common sense and your own security best practices when setting up a system like this on your own.

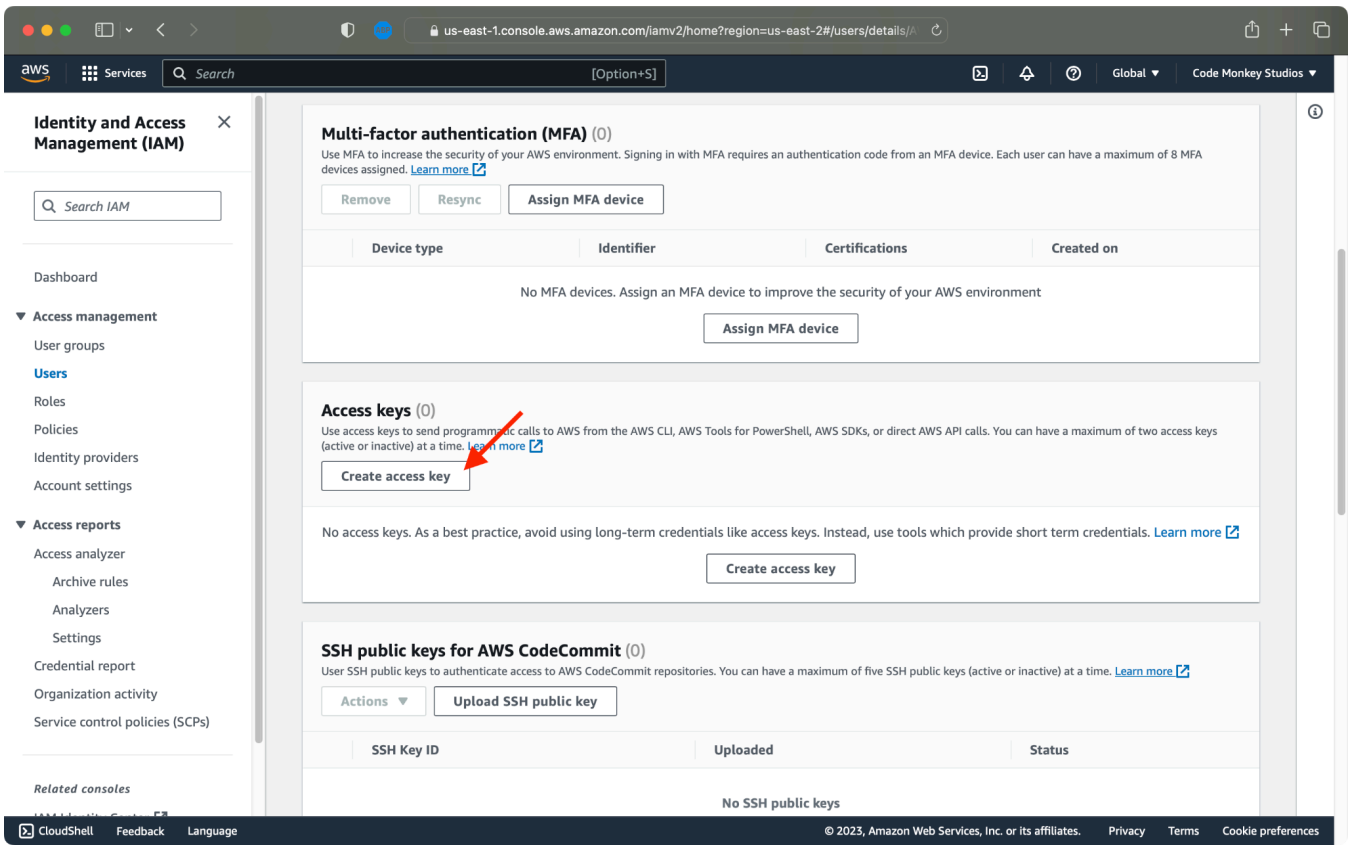
- [ ] Once the user has been created, select the AWSCodeDeployUser to edit their details.



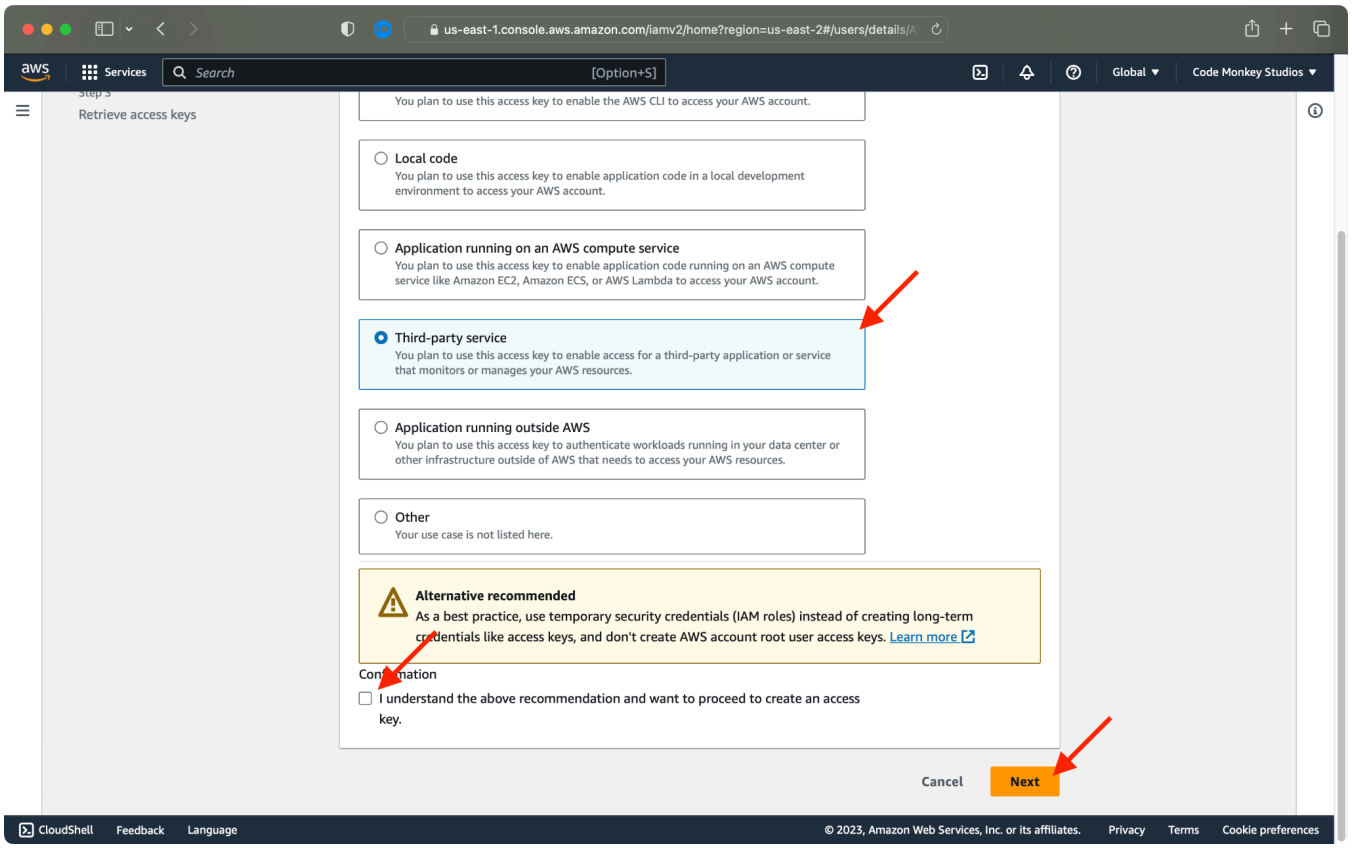

- [ ] Click on the Security Tab in the user details page.



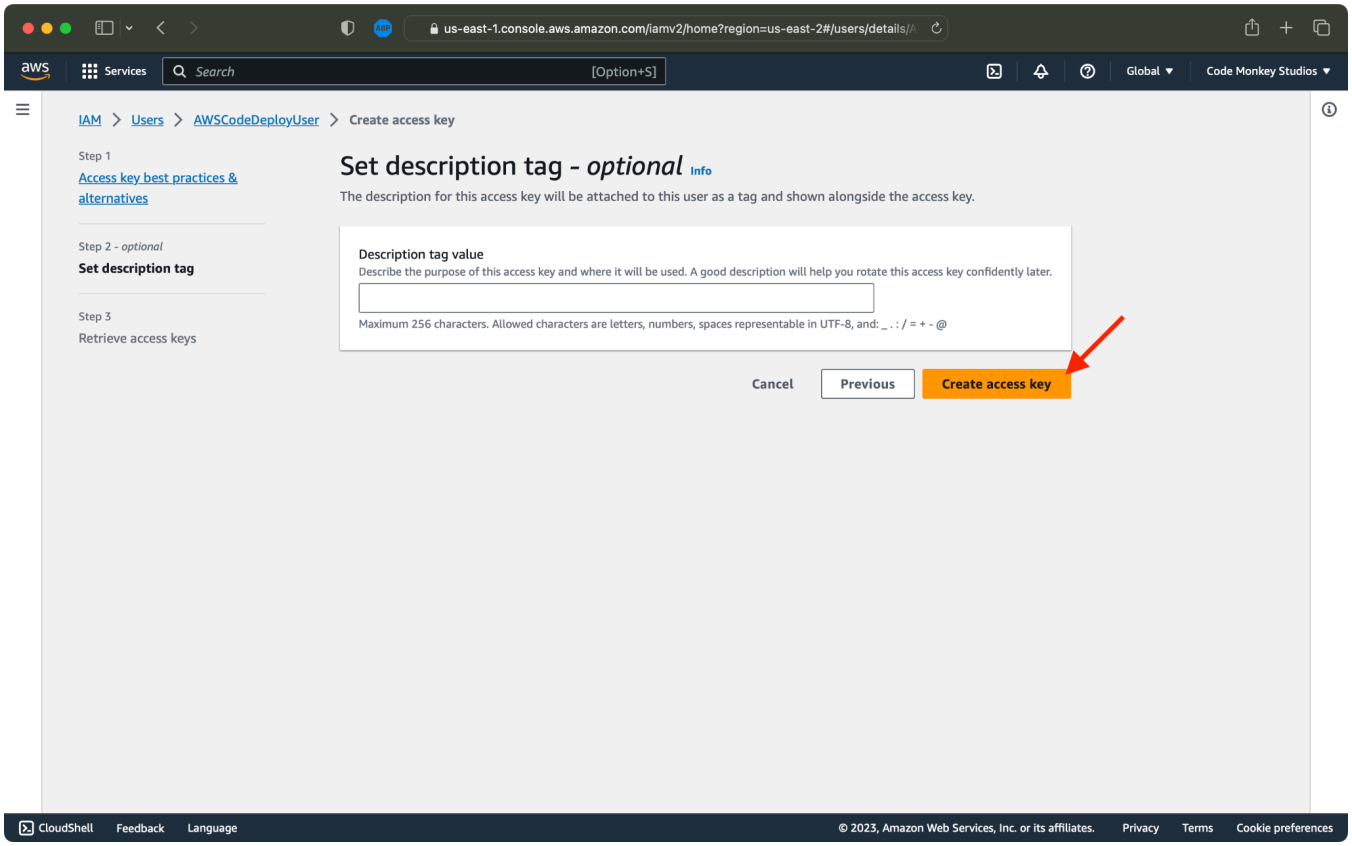

☐ About half way down the security tab, click the Create Access Key button.



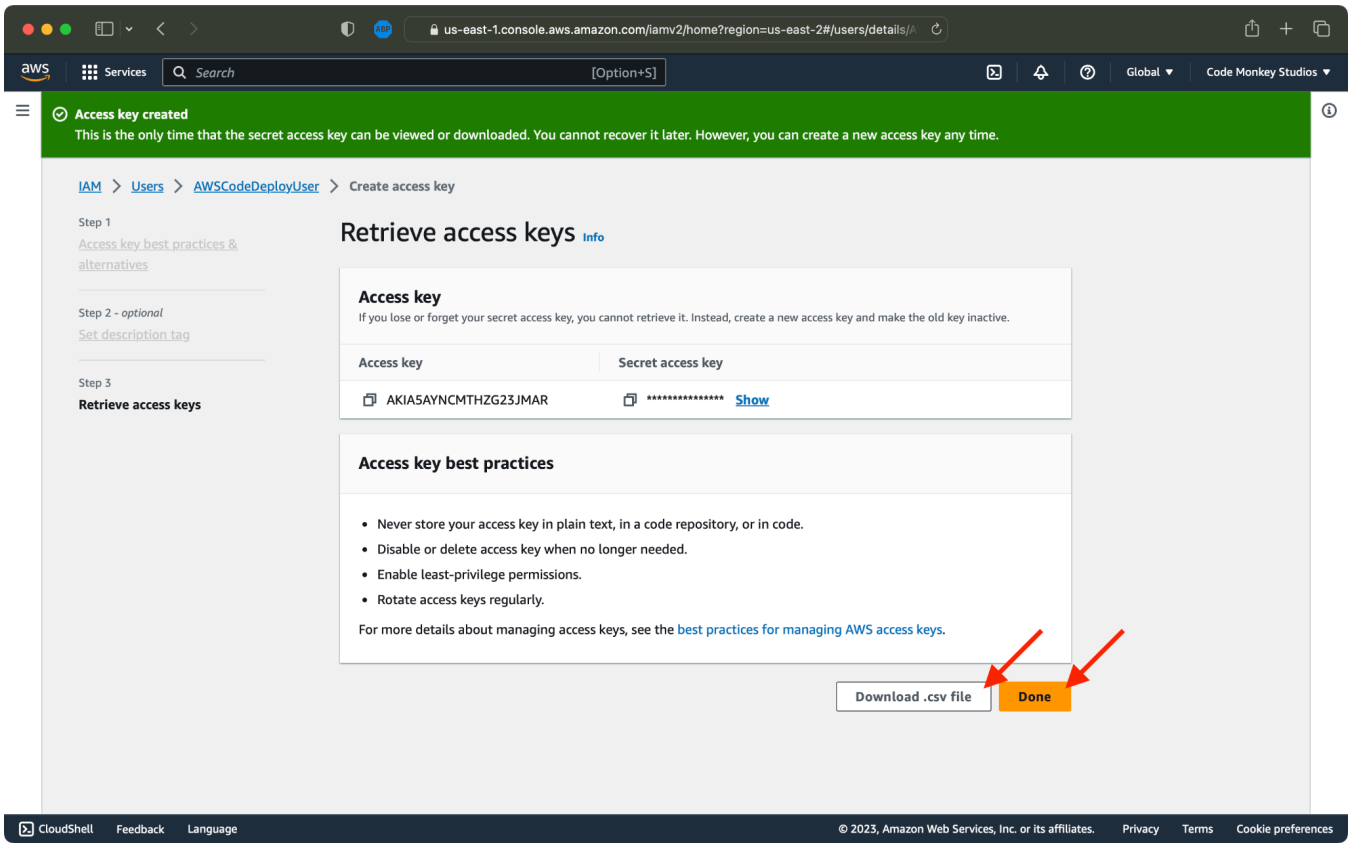
- [ ] On the Access Key Best Practices & Alternatives page, select the Third-Party Service radio button.
- [ ] Check the Confirmation checkbox.
- [ ] Click the Next button at the bottom of the page.




- [ ] On the next screen, set a description tag if you'd like, then click the Create Access Key button at the bottom of the page.




- ☐ From the Retrieve Access Keys page, make sure you have a copy of the Access Key and the Secret Access Key. Click on the Download .csv button to download a file containing your access keys. Do not lose this. If you lose your secret access key, your only option is to revoke the existing key and generate a new one.
- ☐ Once you have a copy of your access keys, click the Done button.

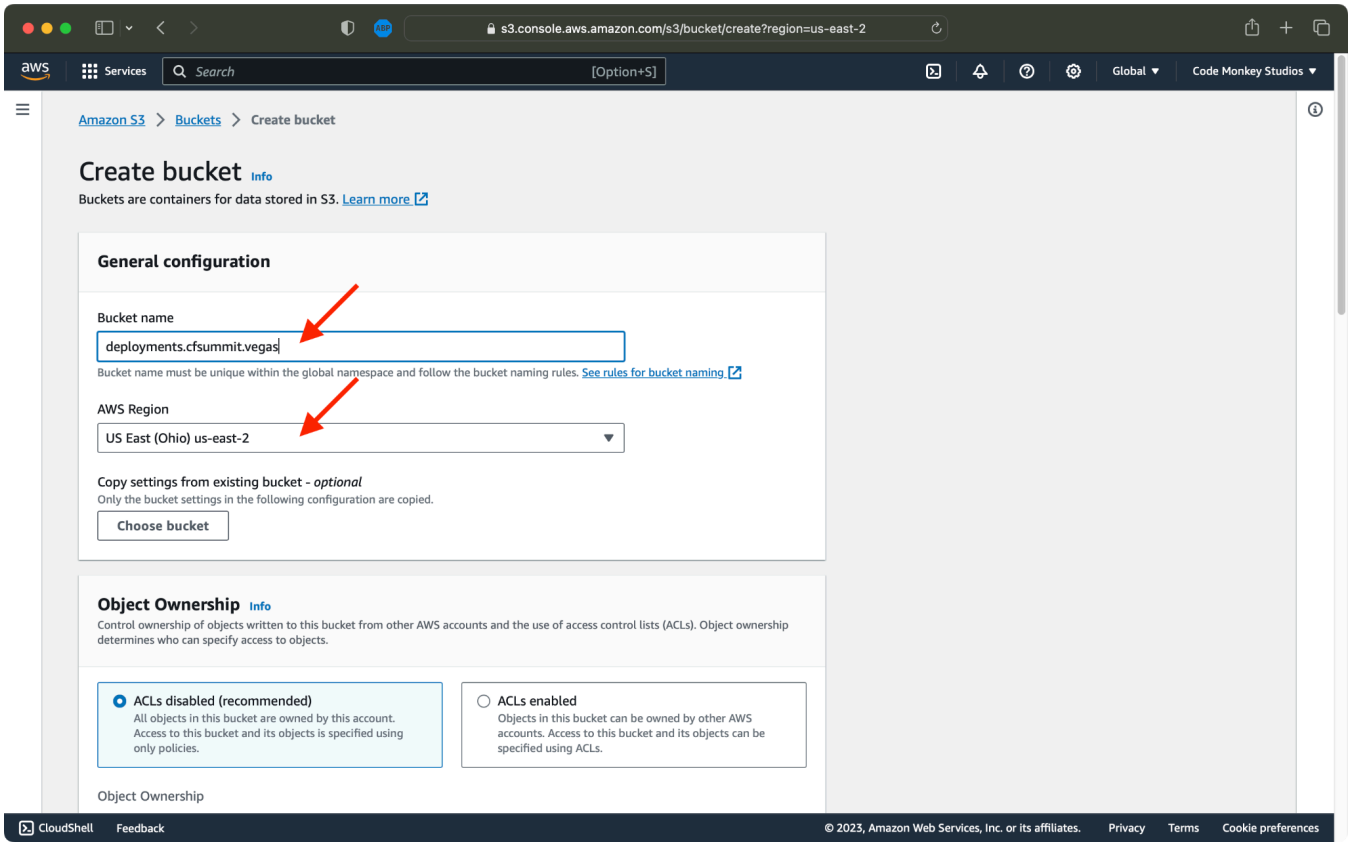



We now have an AWS IAM User, Group, and Role that will be used in BitBucket to perform our Code Deployments.

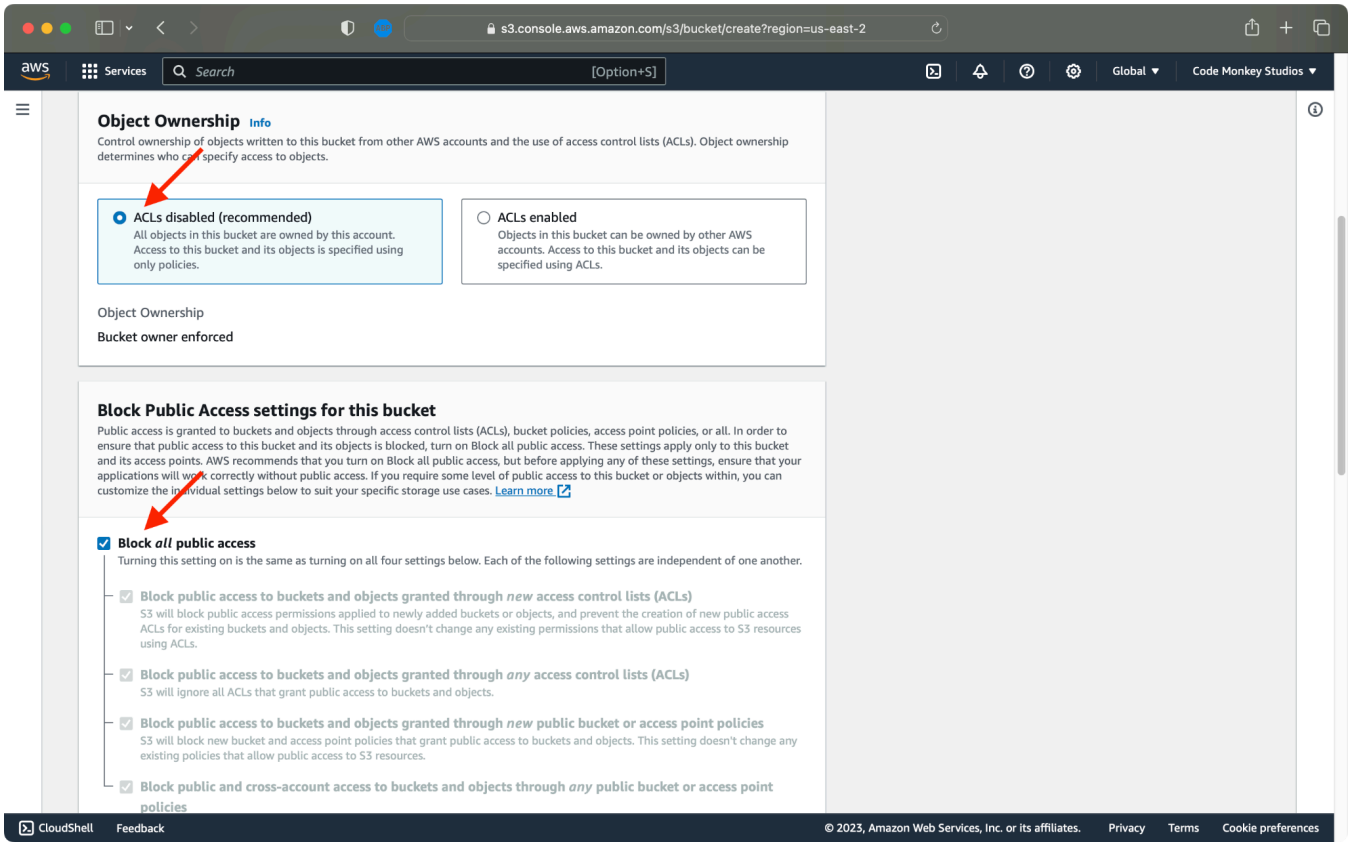
# Part 3 - AWS Simple Storage Service (S3) Setup

> The naming of the bucket here is critical. By default, BitBucket uploads your code to an AWS S3 bucket named {your AWS application name}-codedeploy-deployment. We will create the application later, but for now, I know that the application I am going to create will be called cfsummit-vegas-production.

- [ ] Go to the AWS S3 service.
- [ ] Click the Create Bucket button.
- [ ] On the Create Bucket screen, name the bucket `cfsummit-vegas-production-codedeploy-deployment`.
- [ ] Choose an AWS Region to create the bucket. This should be the same region you intend to create your AWS EC2 instances in. For this demonstration, I'm going to choose us-east-2 (Ohio).



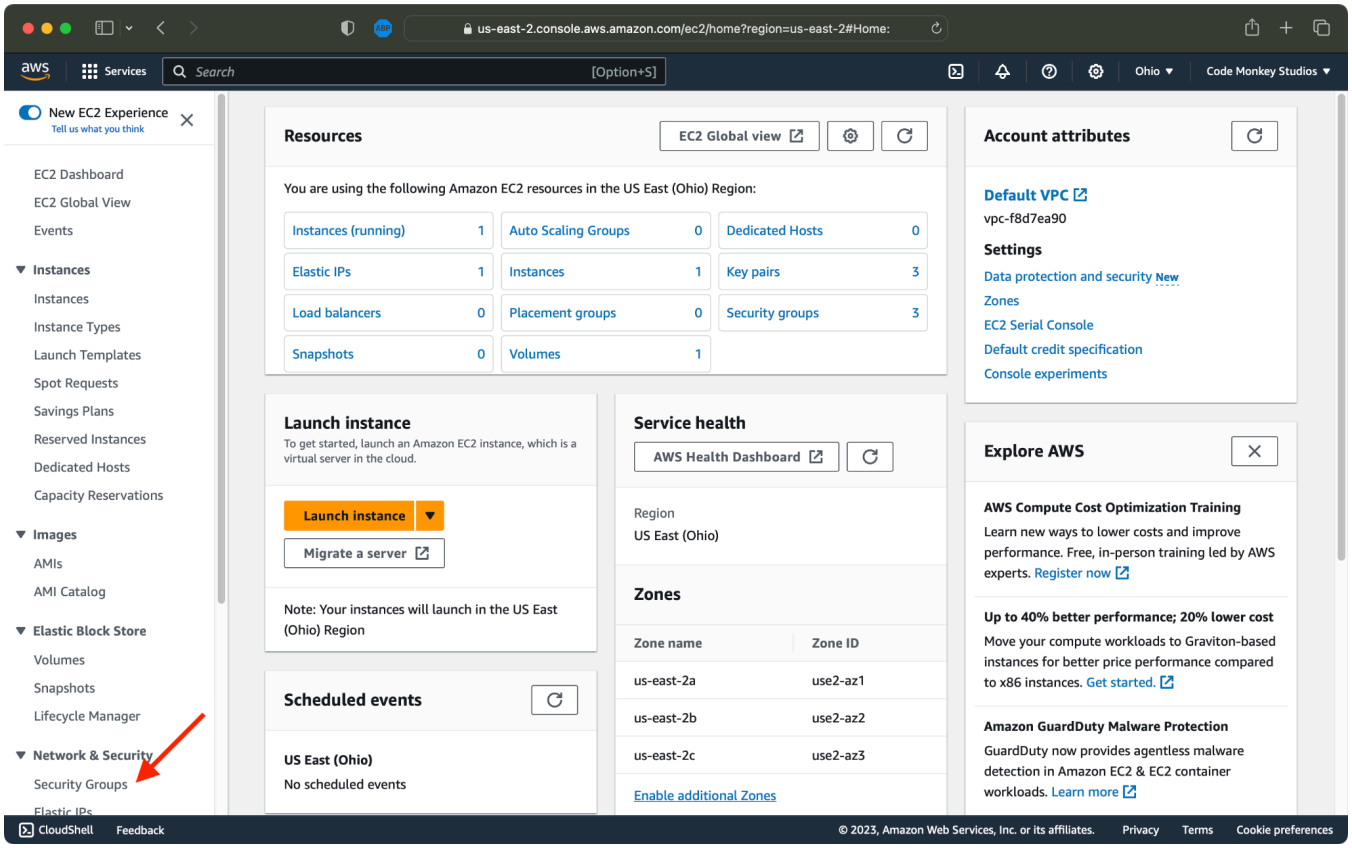

- ☐ On the same page, make sure ACL's Disabled is selected.
- ☐ Make sure Block All Public Access is checked.
- ☐ Click the Create Bucket button at the bottom of the page.



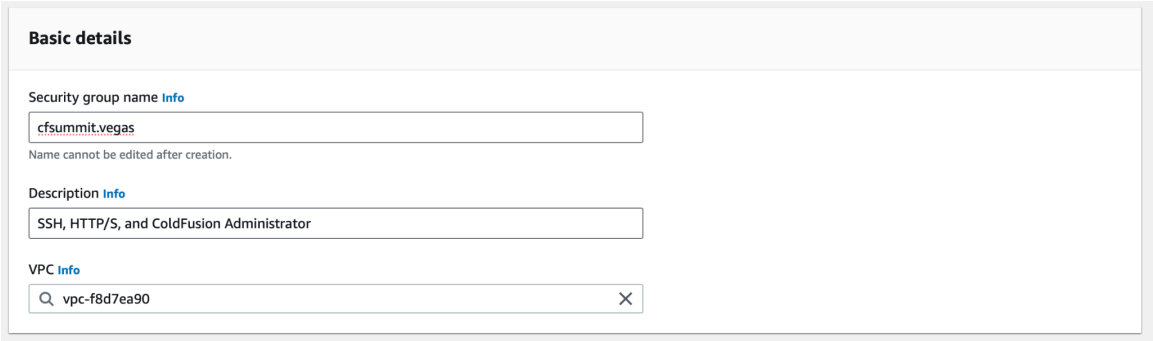

# Part 4 - Create the AWS EC2 Instance

## Step 1 - Create and Configure a Security Group

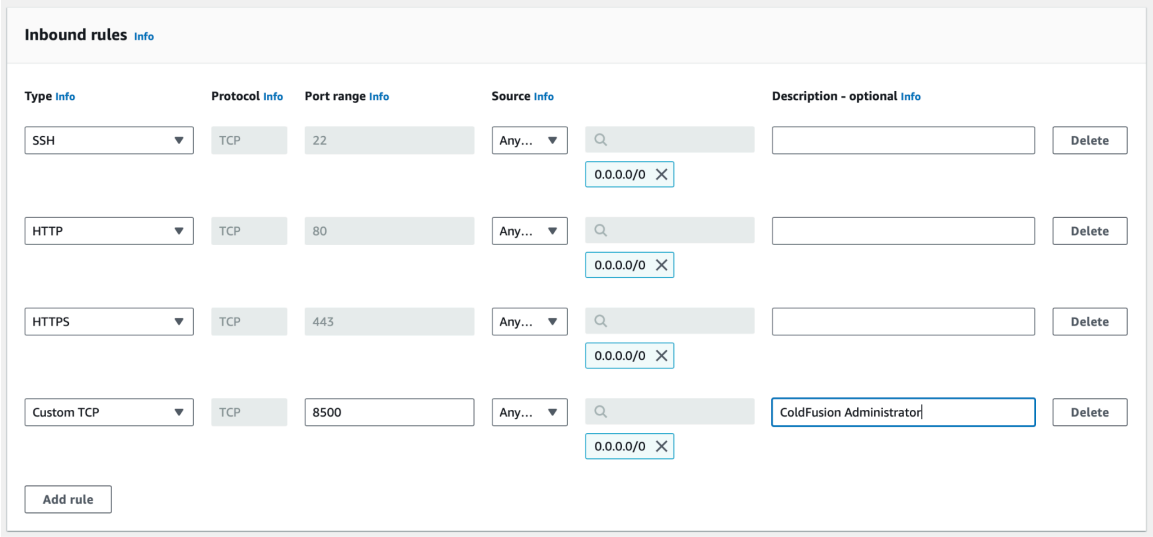
- ☐ Go to the AWS EC2 Service.
- ☐ From the left column, click the link for Security Groups




- ☐ On the next page, click the Create Security Group button.
- ☐ In the Basic Details section:
  - ☐ Name the Security Group `cfsummit.vegas`.
  - ☐ Set the Description as `SSH, HTTP/S, and ColdFusion Administrator`.




- ☐ In the Inbound Rules section, we are going to create four separate inbound traffic rules:
  - ☐ Type: SSH, Source: My IP
  - ☐ Type: HTTP, Source: Anywhere-IPv4
  - ☐ Type: HTTPS, Source: Anywhere-IPv4
  - ☐ Type: Custom TCP, Port Range: 8500, Source: My IP, Description: ColdFusion Administrator

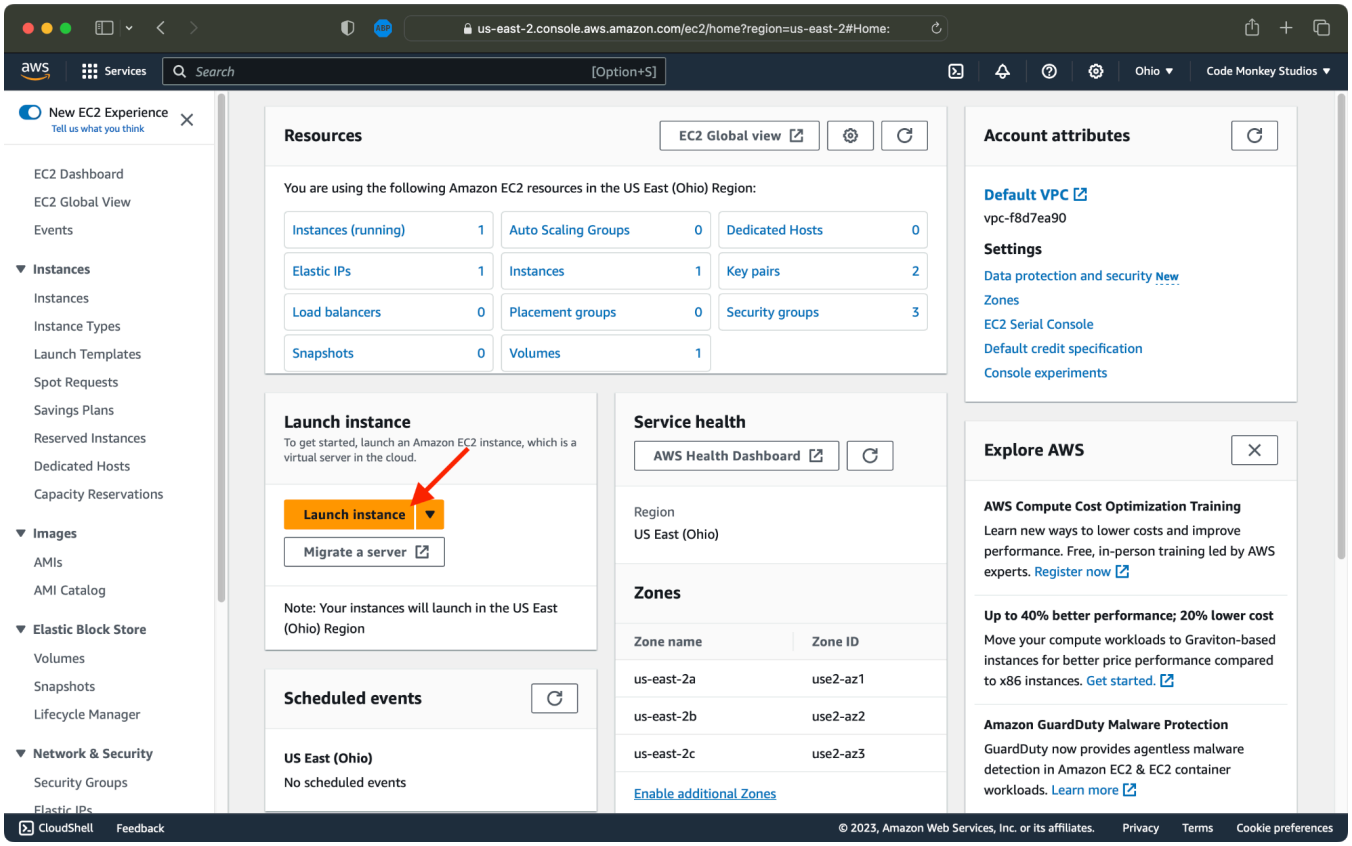



Remember... security? Out the window. You would never want to leave these security rules like this.

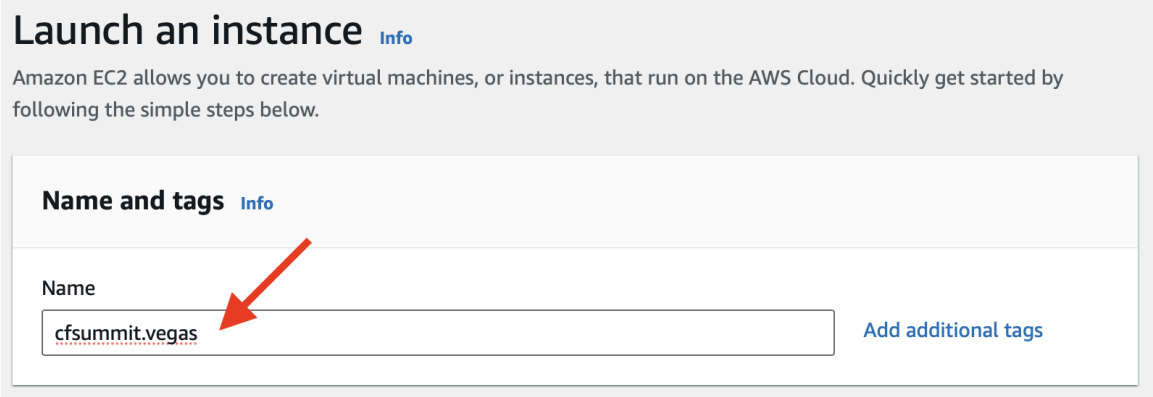
- ☐ Click the Create Security Group button at the bottom of the window.

## Step 2 - Configure and Launch the EC2 Instance

- ☐ Go to the AWS EC2 Service.
- ☐ Click the Launch Instance button.

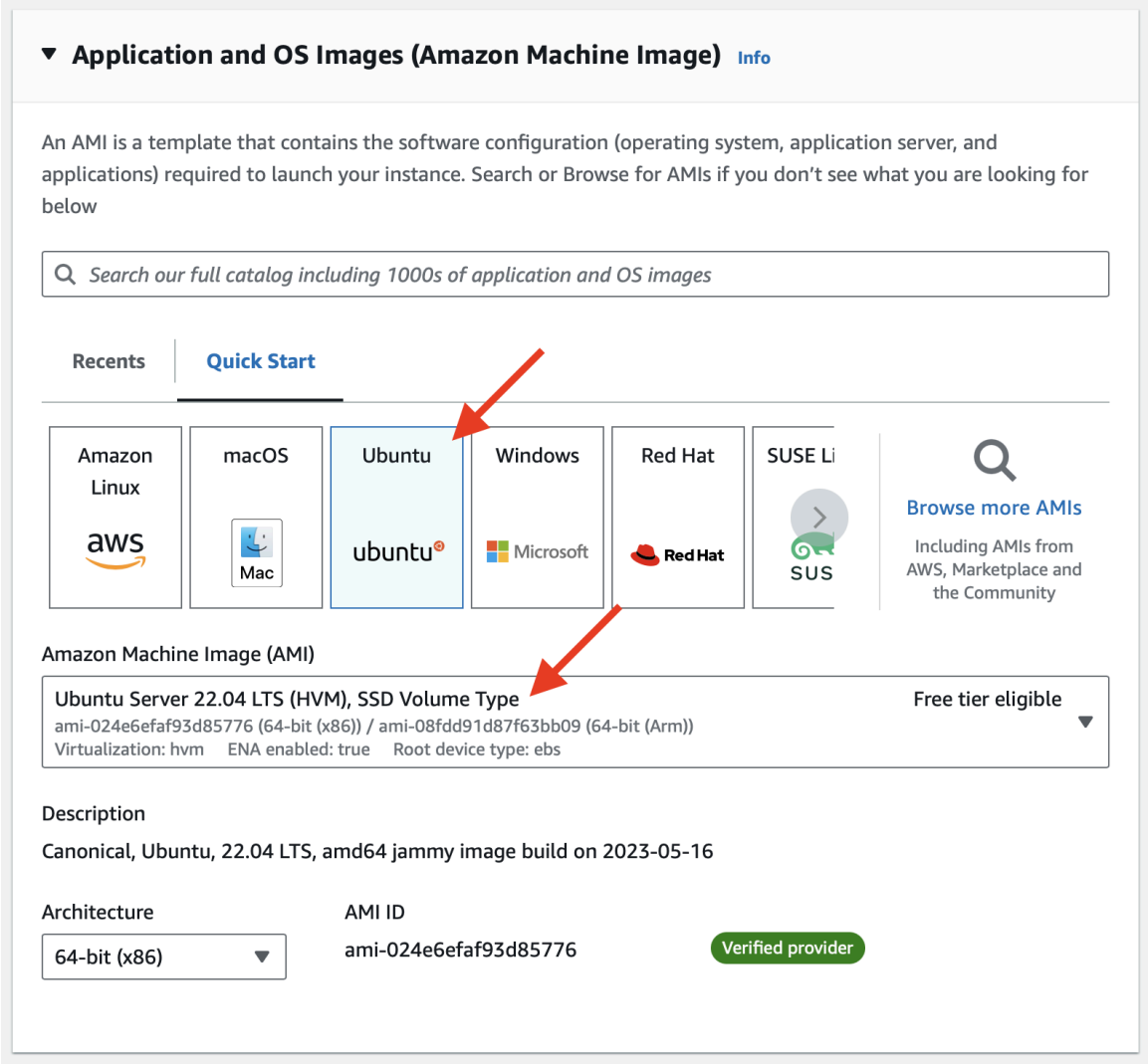



☐ On the Launch an Instance page, name the instance `cfsummit.vegas`.

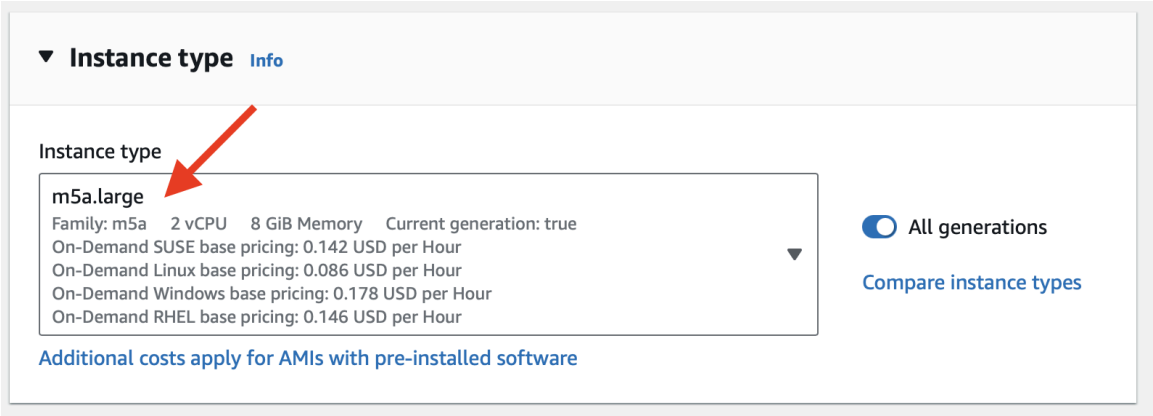



☐ In the Application and OS Images section:

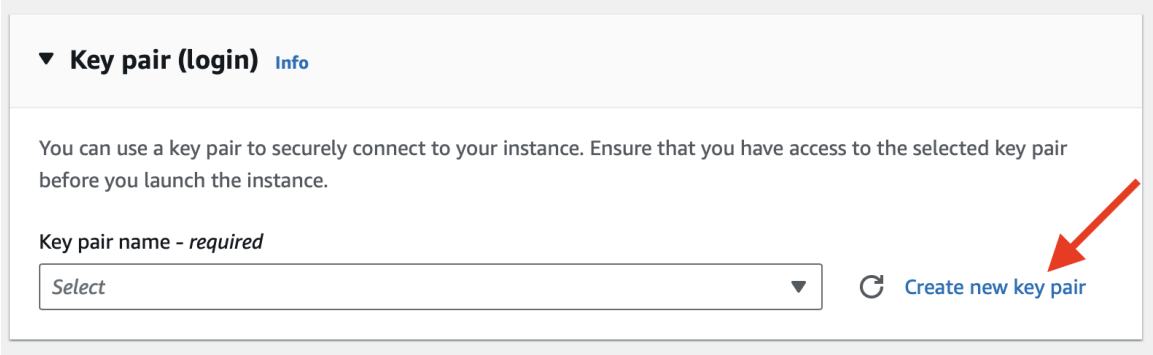
- ☐ Select Ubuntu
- ☐ Select Ubuntu 22.04



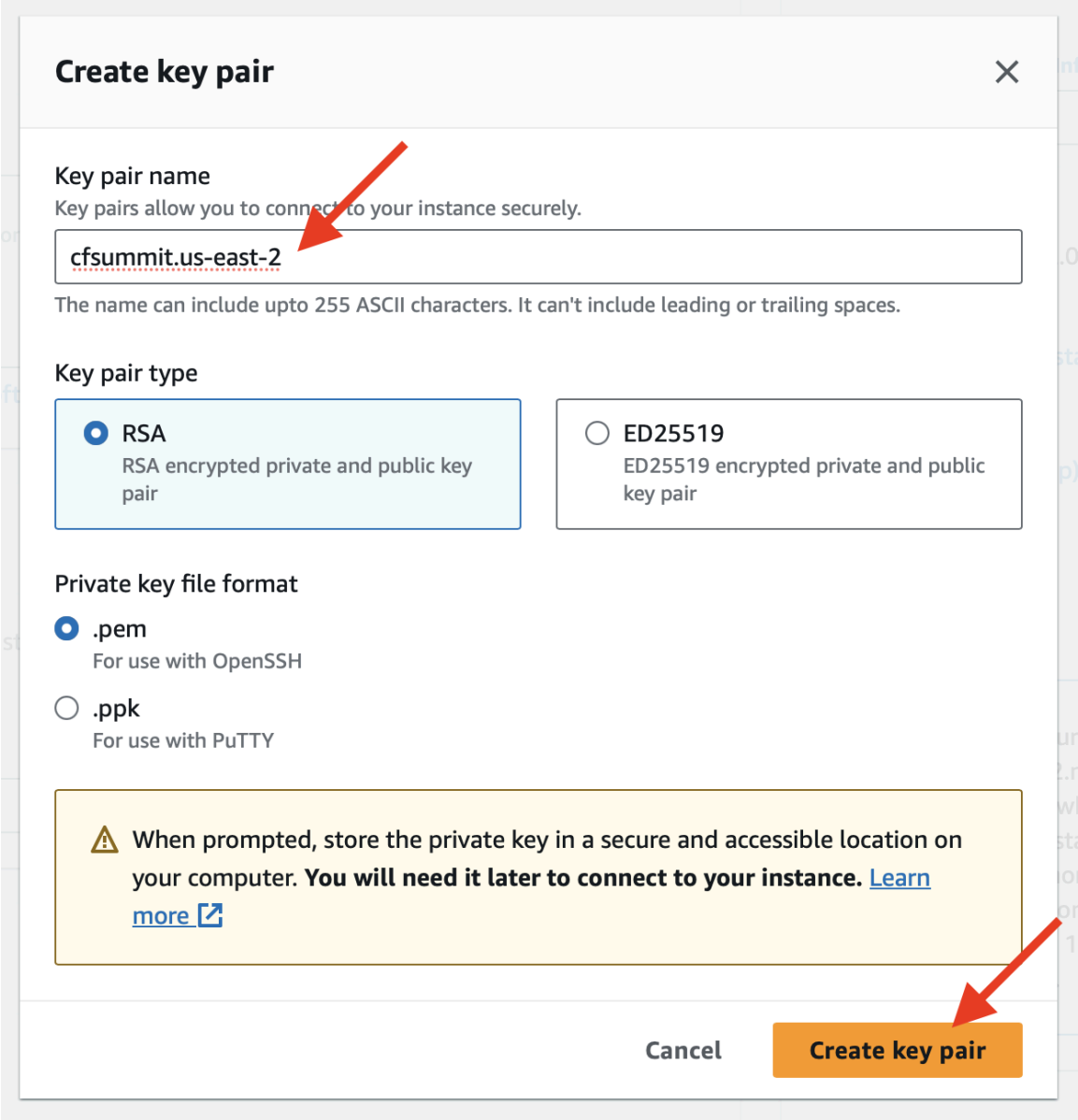

☐ In the Instance Type section, select m6a.large



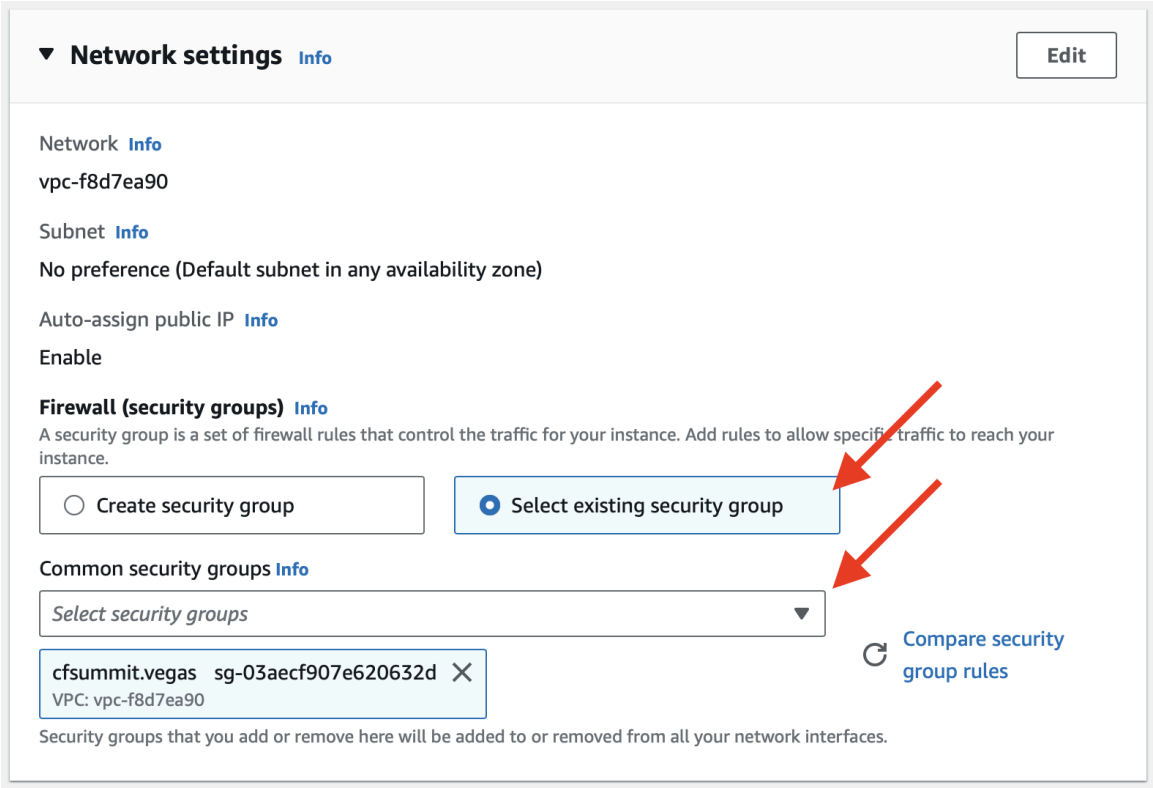

☐ In the Key Pair section, click the "Create a New Key Pair" link.



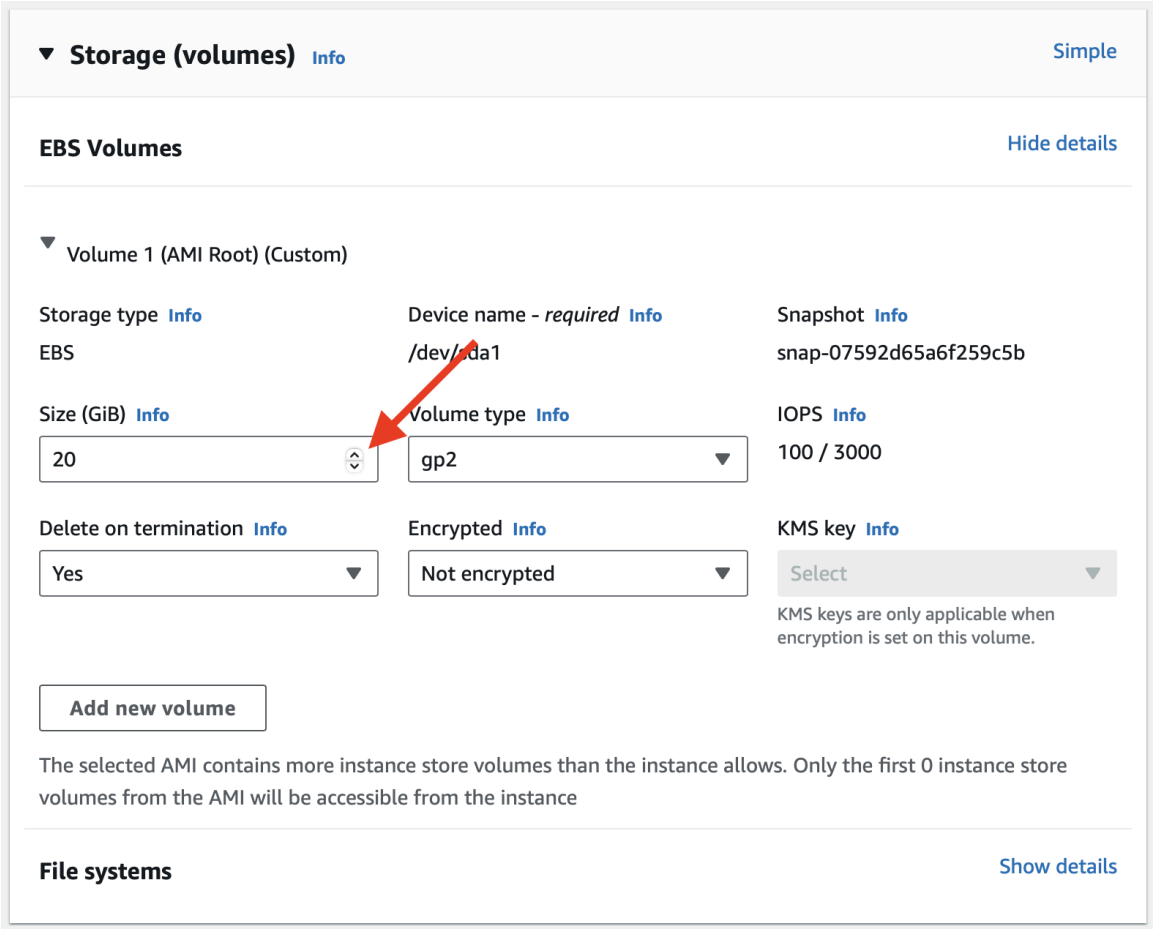

- ☐ In the Create Key Pair modal:
  - ☐ Name the key pair cfsummit.us-east-2
  - ☐ Leave the key pair type as RSA and the private key file format as .pem.
  - ☐ Click the Create Key Pair button at the bottom of the modal window. This will download the .pem file to your local computer.



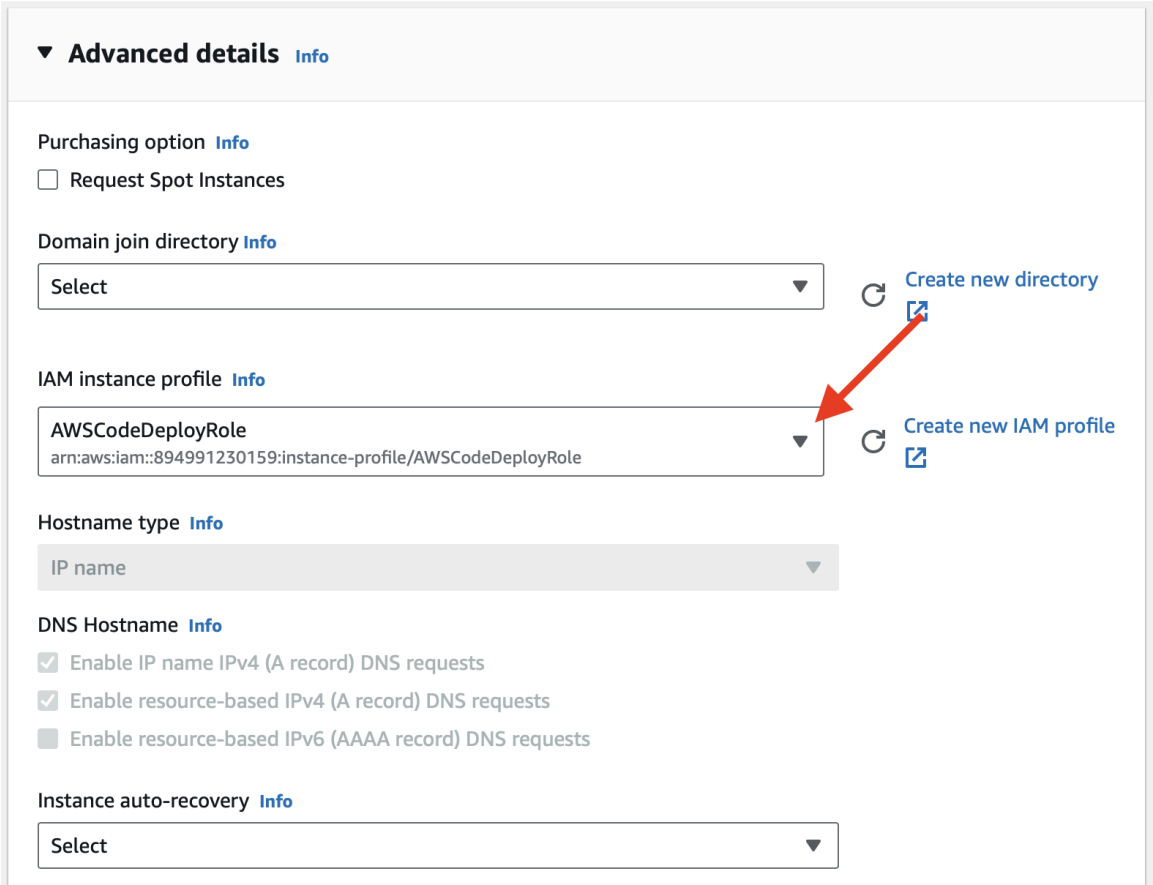
- [ ] In the Network Settings section:
  - [ ] Select the Select Existing Security Group radio button.
  - [ ] Click the dropdown to select the cfsummit.vegas security group we created in Step 1.



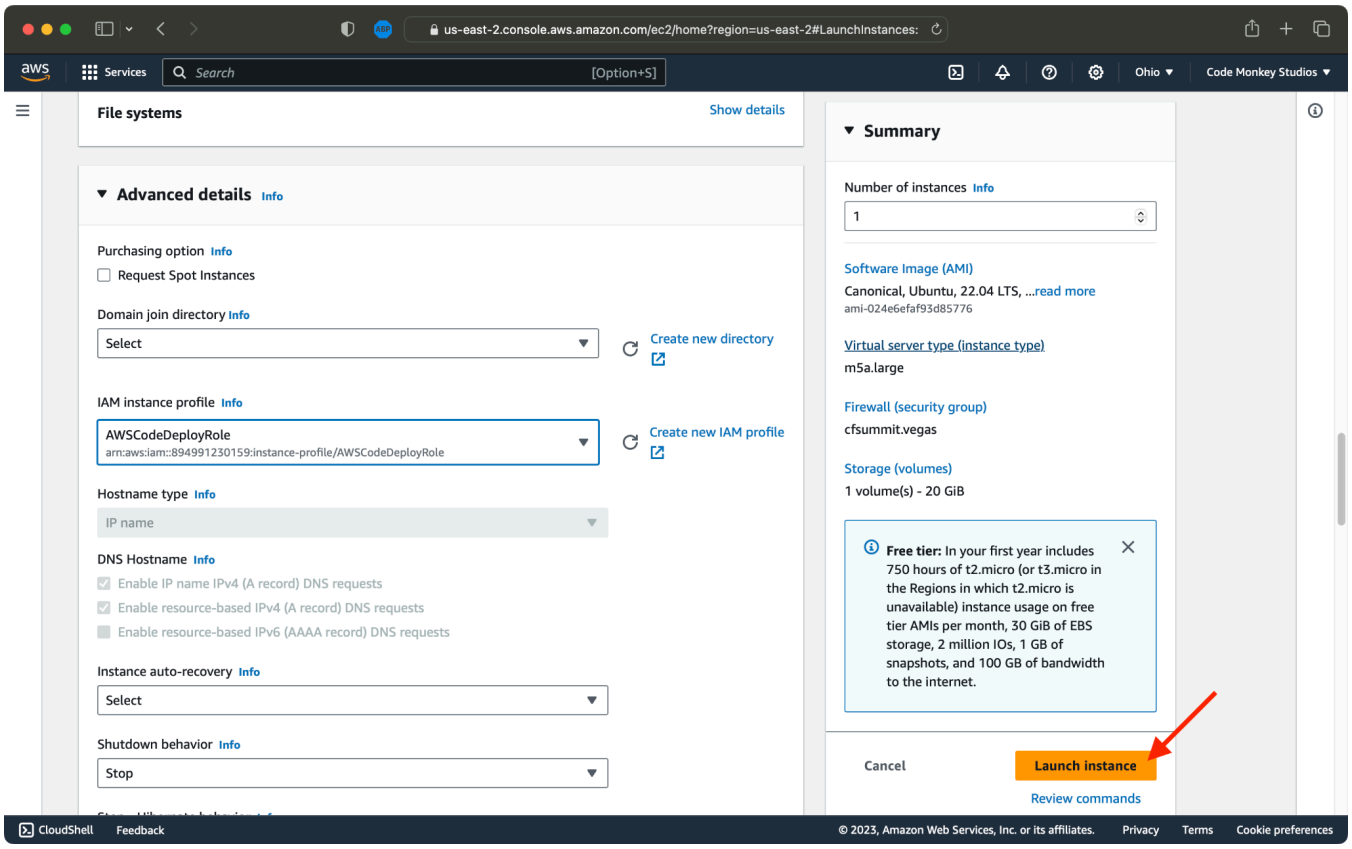

- [ ] In the Storage (Volumes) section:
  - [ ] Increase the storage size to 20 GB.




- [ ] Expand the Advanced Details section.
- [ ] From the IAM Instance Profile dropdown, select the AWSCodeDeployRole role we created earlier.

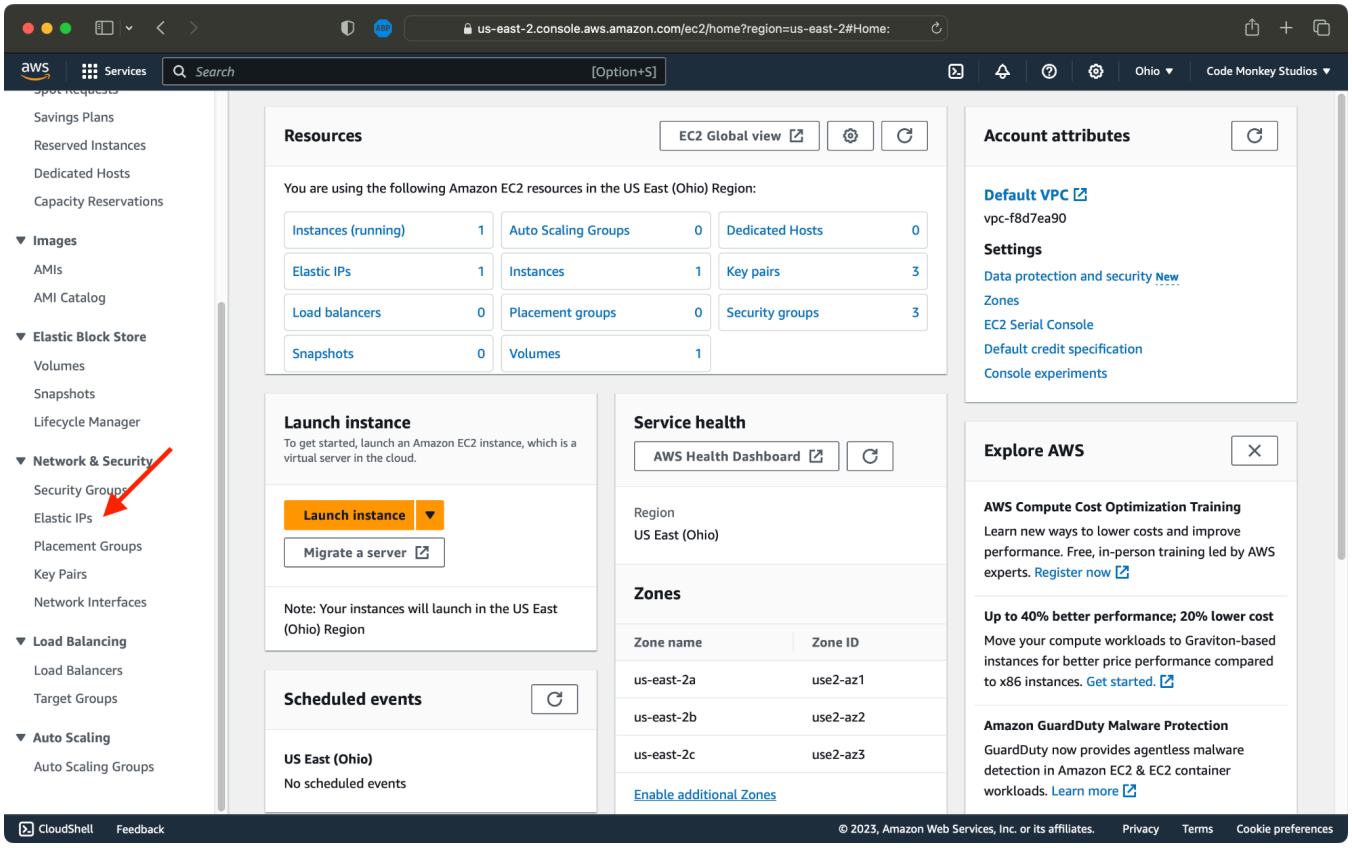



☐ Finally, click the Launch Instance button in the right column, near the bottom of the page.

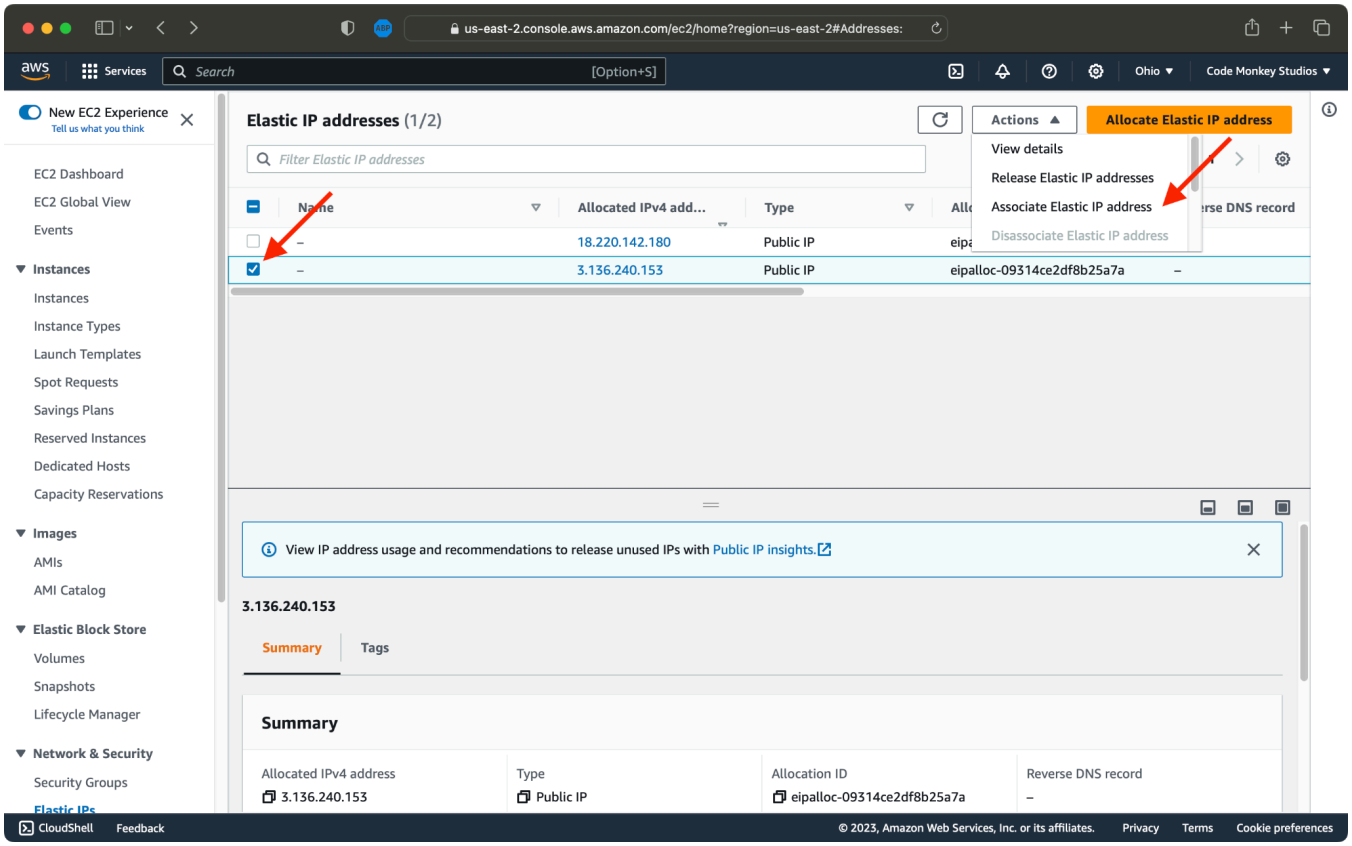


# Step 3 - Assign an Elastic IP Address to the EC2 Instance

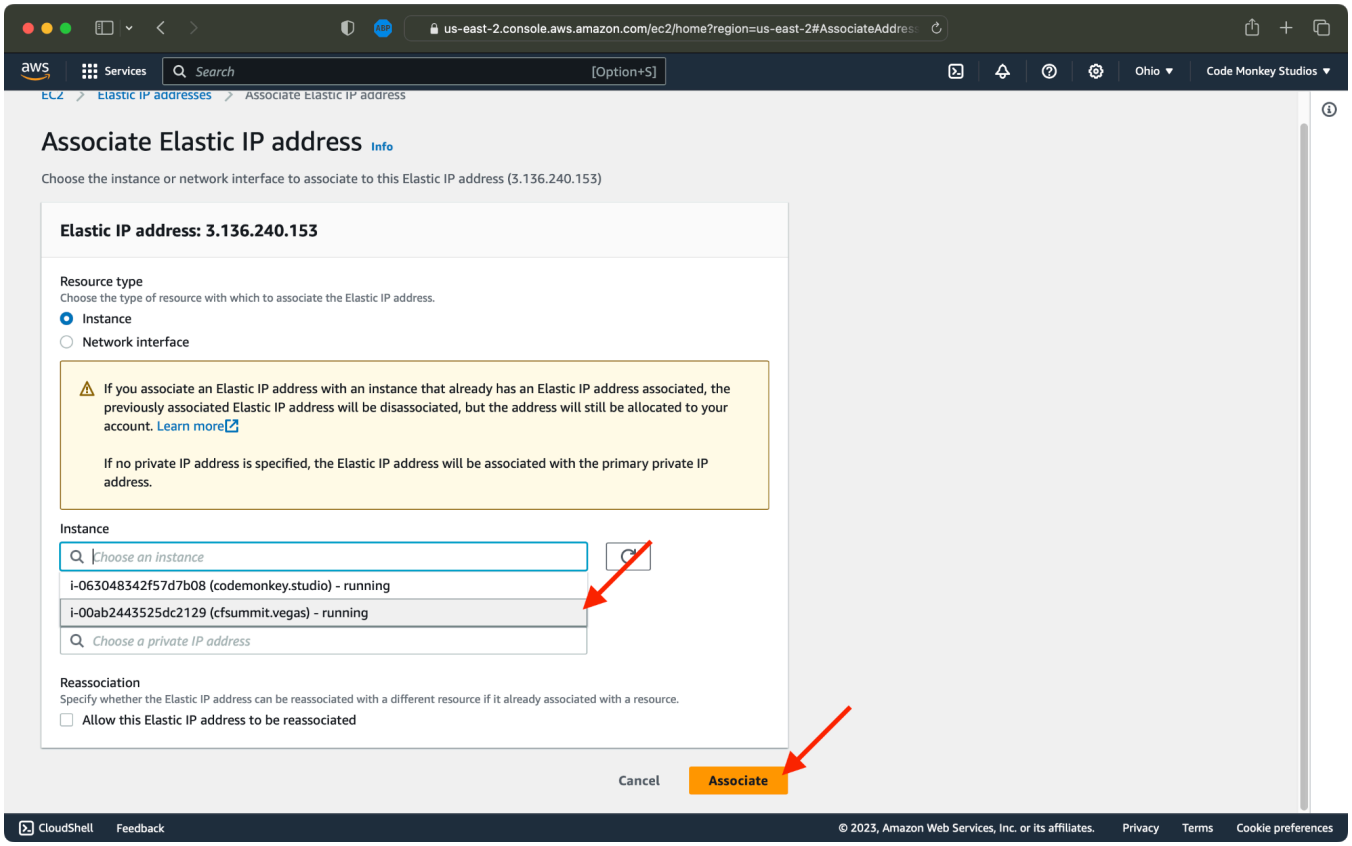
- [ ] Go to the AWS EC2 Service.
- [ ] In the left column, select the link labeled Elastic IPs.




- [ ] Click the Allocate Elastic IP Address button.
- [ ] On the next page, leave everything default, and click the Allocate button at the bottom of the page.
- [ ] Copy the newly created IP Address.
- [ ] Return to the Elastic IP Addresses page.
- [ ] Check the checkbox next to the newly created IP Address.
- [ ] From the Actions dropdown menu at the top of the page, click the the Associate Elastic IP Address button.



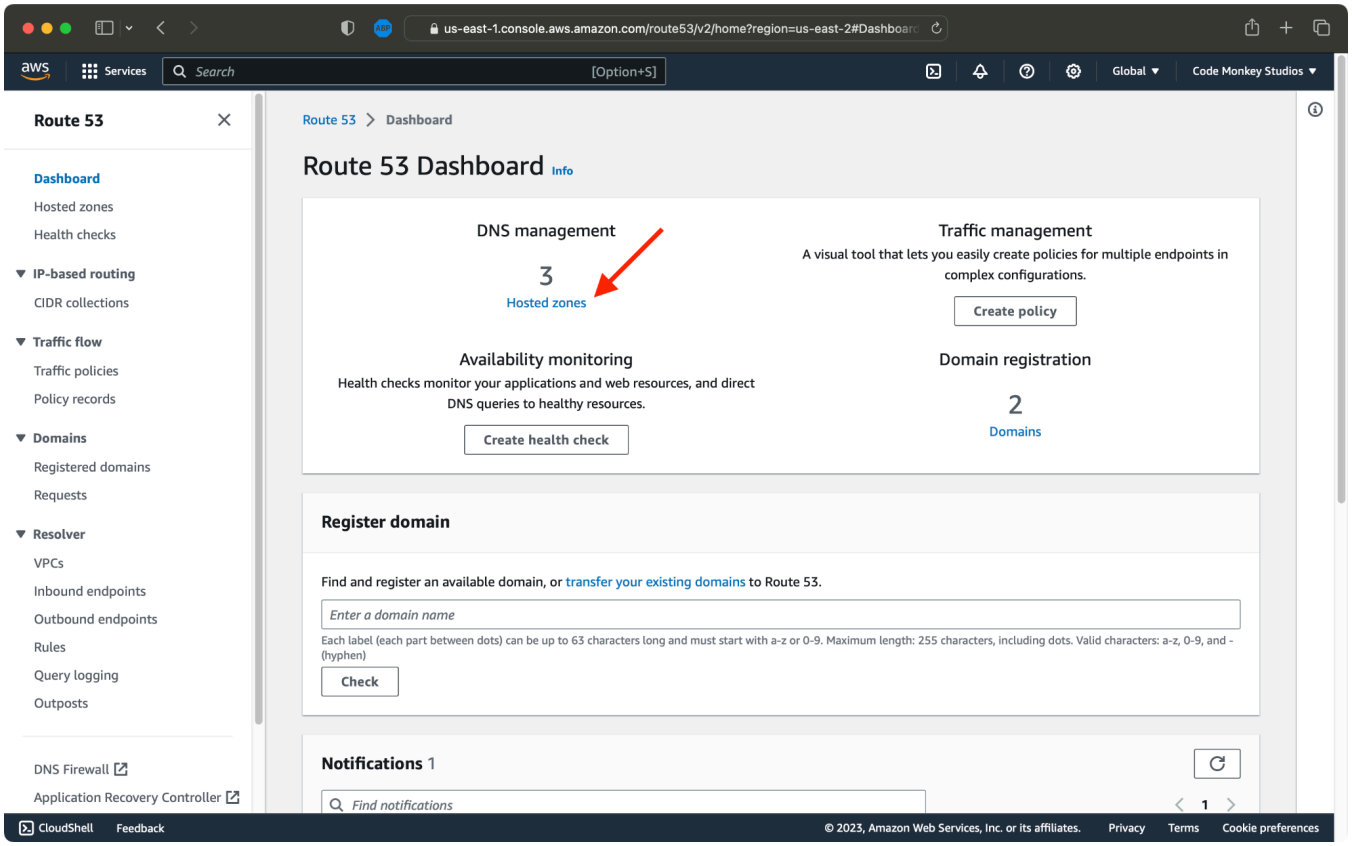

- ☐ In the Associate Elastic IP address page:
  - ☐ Expand the Instance dropdown.
  - ☐ Select the cfsummit.vegas instance.
  - ☐ Click the Associate button at the bottom of the page.



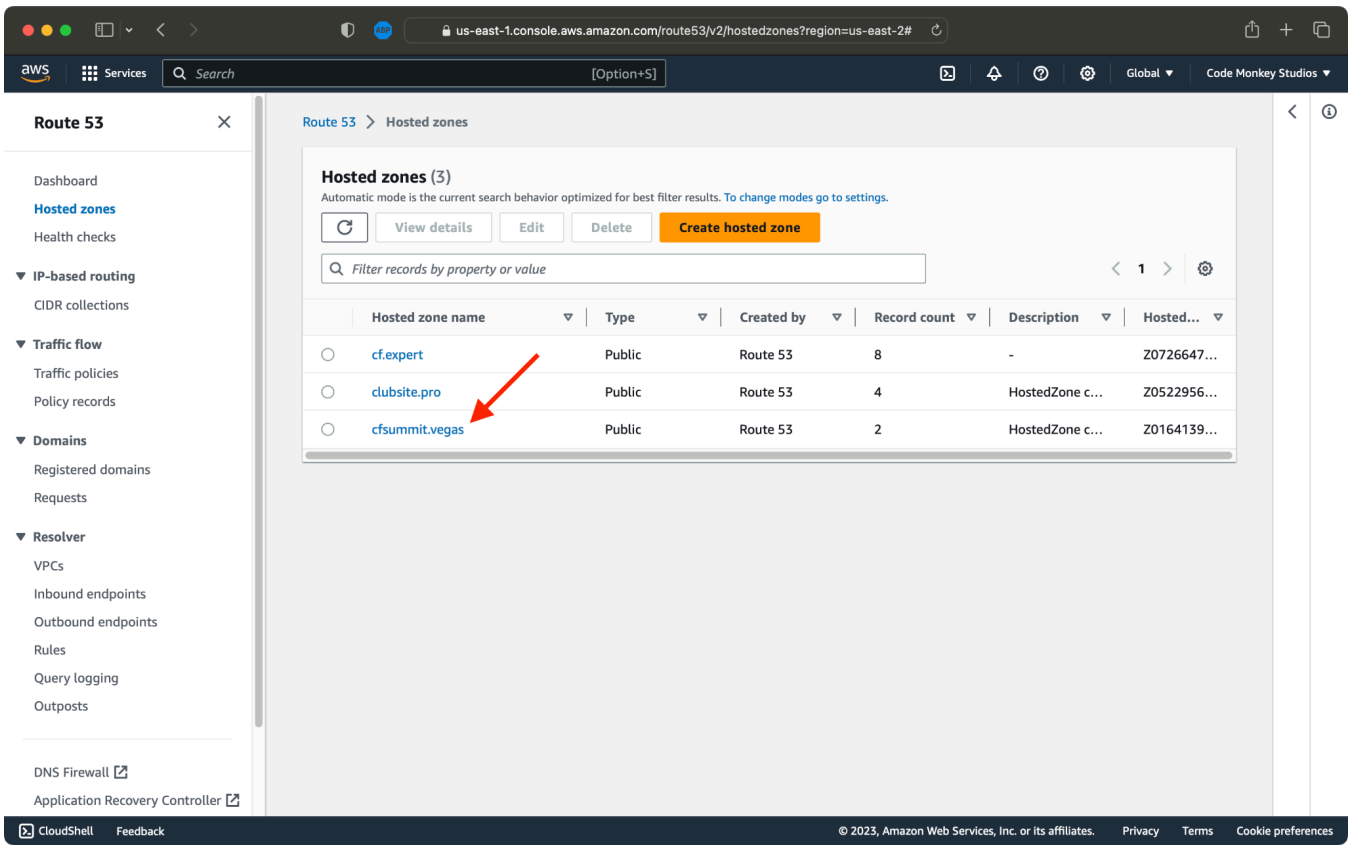

The Elastic IP Address is now associated with the EC2 instance and traffic to that IP will be directed at that server.

## Step 4 - Update the DNS

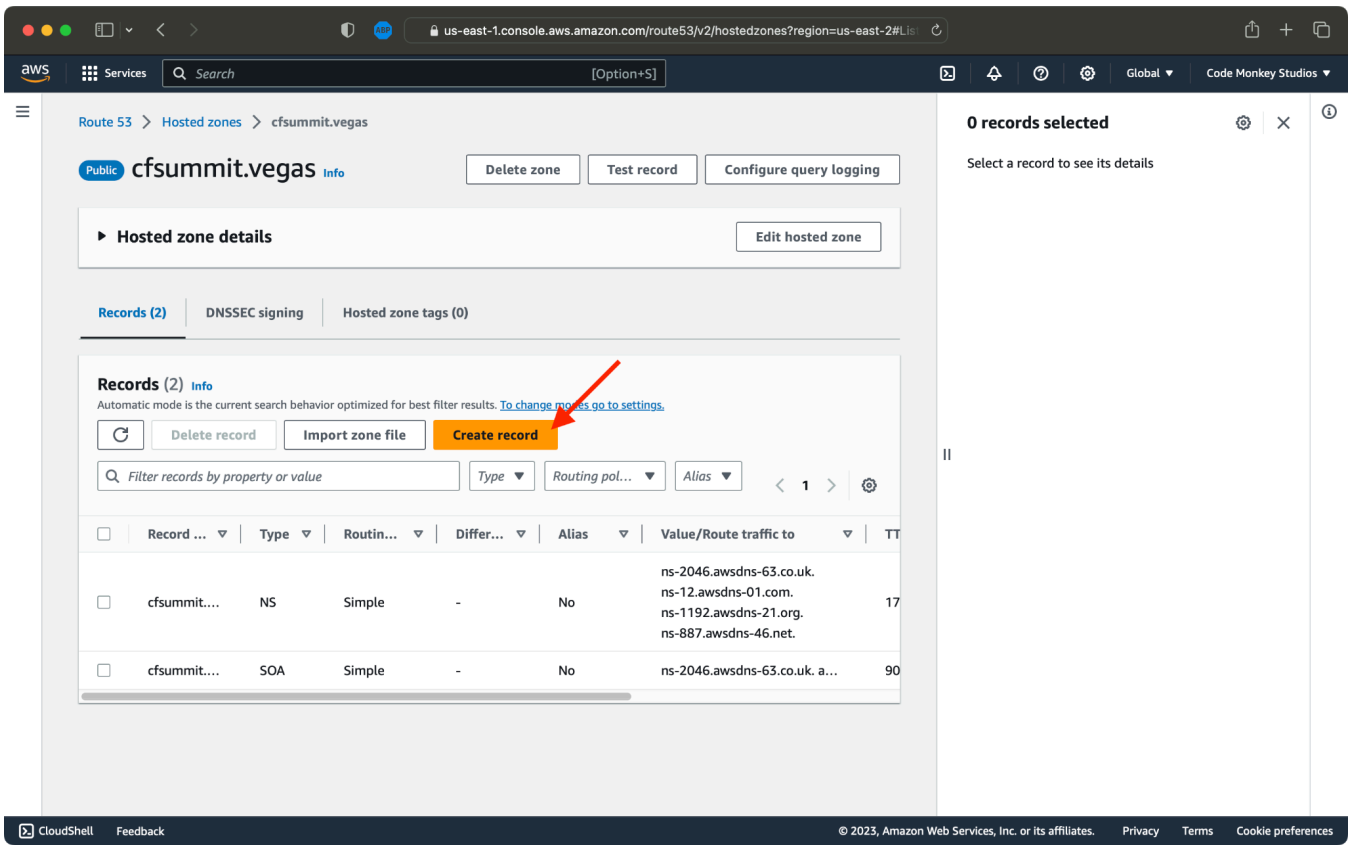
- [ ] Go to the AWS Route 53 Service.
- [ ] Select the link labeled Hosted Zones



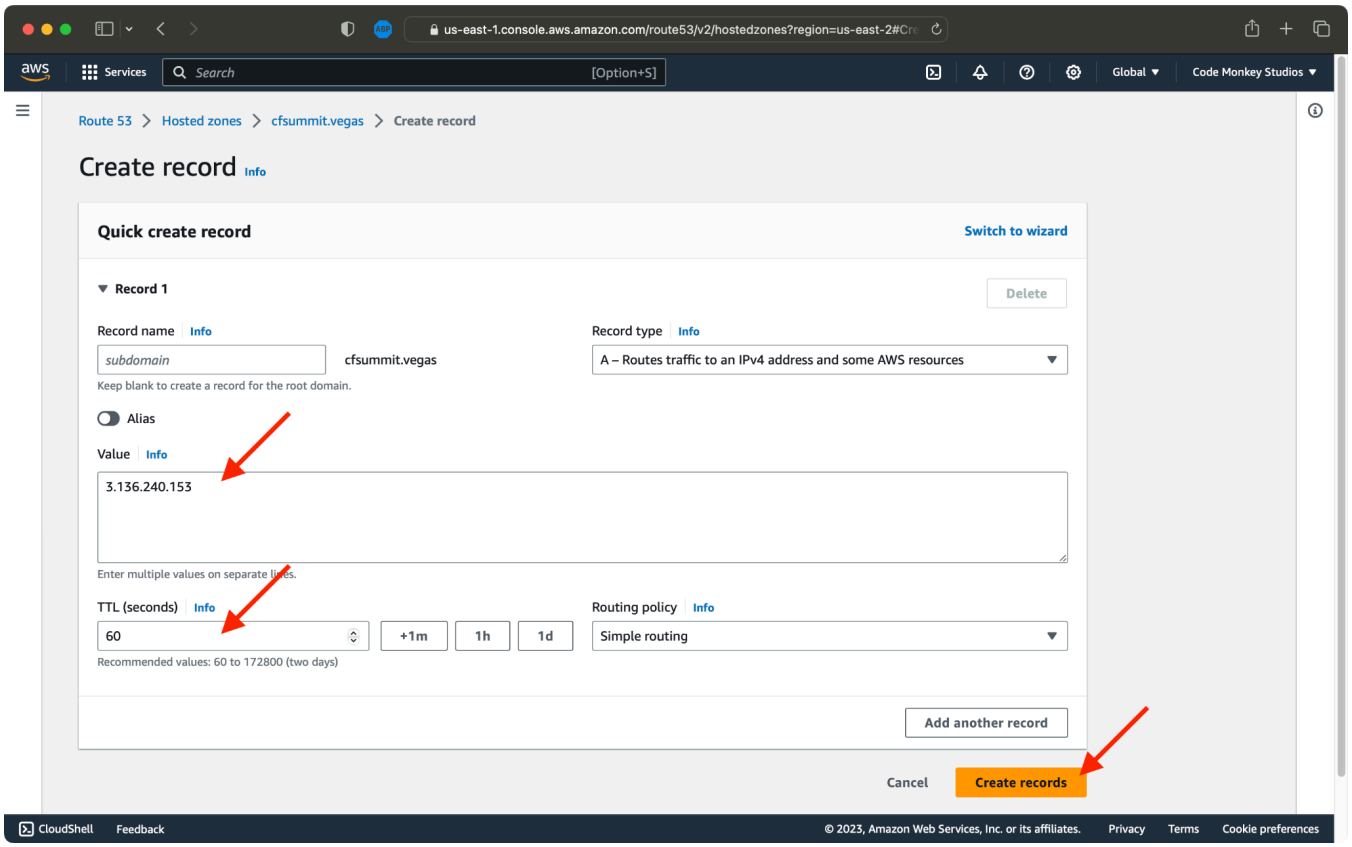

☐ Select the cfsummit.vegas hosted zone.



- [ ] Select the Create Record button.




- [ ] From the Create Record page:
    - [ ] Paste the IP Address we created in the previous step into the Value field.
    - [ ] Set the TTL to one minute.
    - [ ] Click the Create Records button at the bottom of the page.




Traffic addressed to cfsummit.vegas will now be routed to the Elastic IP Address.

## Step 5 - Connect to the EC2 Instance

- ☐ From the local computer, open the Terminal application.
- ☐ First we will change the permissions on the downloaded cfsummit.vegas.us-east-2.pem file by running the following command:

```
chmod 400 ~/Downloads/cfsummit.us-east-2.pem
```

- ☐ Copy the file to the local computer's SSH folder, which is `~/.ssh` on a Mac, by running the following command.

```
cp ~/Downloads/cfsummit.us-east-2.pem ~/.ssh
```

- ☐ Finally, connect to the EC2 Instance by running the following command:

```
ssh -i ~/.ssh/cfsummit.us-east-2.pem ubuntu@cfsummit.vegas
```

- [ ] Once connected, it's helpful to log in as a root account so we do not have to continuously run `sudo` at the beginning of every command.

```
sudo -s
```

We are now logged into the AWS EC2 instance with superuser privileges.

## Step 6 - Install Server Software

### Create a Maintenance Update Script

- [ ] Run the following commands:

```
cd /home/ubuntu
touch update-server
chmod +x update-server
nano update-server
```

- [ ] Paste the following:

```
#!/bin/bash
apt update && apt upgrade -y
```

- [ ] Press Ctrl-O, then Enter to save the file.
- [ ] Press Ctrl-X to exit nano.
- [ ] Run the following commands:

```
chmod +x update-server
./update-server
reboot now
```

### Apache 2

- [ ] Run the following command:

```
apt install -y apache2
```

- [ ] Once Apache is installed, run the following commands:

```
a2dissite 000-default
rm -rf /var/www/html
mkdir /var/www/cfsummit.vegas
cd /etc/apache2/sites-available
touch cfsummit.vegas.conf
nano cfsummit.vegas.conf
```

- [ ] Paste the following:

```
<VirtualHost *:80>

     ServerName cfsummit.vegas
     ServerAdmin david@codemonkey.studio
     DocumentRoot /var/www/cfsummit.vegas/
     DirectoryIndex index.cfm

     <Directory /var/www/cfsummit.vegas>
          Options Indexes FollowSymLinks
          AllowOverride All
          Order allow,deny
          Allow from all
     </Directory>

     ErrorLog ${APACHE_LOG_DIR}/cfsummit.vegas.error.log
     CustomLog ${APACHE_LOG_DIR}/cfsummit.vegas.access.log combined

</VirtualHost>
```

- [ ] Press Ctrl-O, then Enter to save the file.
- [ ] Press Ctrl-X to exit nano.
- [ ] Run the following commands:

```
a2ensite cfsummit.vegas
systemctl reload apache2
cd /var/www/cfsummit.vegas
touch index.cfm
nano index.cfm
```

- [ ] Paste the same basic index HTML we created in our local development environment.

```
<h2>cfsummit.vegas</h2>
```

```
<cfoutput>
     <p>#server.coldfusion.productName# (#server.coldfusion.productLevel#)
version #server.coldfusion.productVersion#</p>
     <p>#dateTimeFormat( now(), 'long' )#</p>
</cfoutput>
```

- [ ] Press Ctrl-O, then Enter to save the file.
- [ ] Press Ctrl-X to exit nano.

At this point, we should be able to browse to cfsummit.vegas and see some kind of result.

## CertBot for SSL Certificates

- [ ] To install CertBot, Run the following command:

```
cd /home/ubuntu
snap install core; snap refresh core; snap install --classic certbot; ln -s
/snap/bin/certbot /usr/bin/certbot
```

- [ ] To install an SSL certificate for the website, run the following command:

```
certbot --apache
```

- [ ] Once certbot loads, follow the prompts and establish an SSL cert for the site.

## AWS Code Deploy

- [ ] To install the AWS CodeDeploy Agent on the EC2 instance, run the following commands:

```
cd /home/ubuntu
snap refresh; snap install ruby --classic --channel=2.7/stable
wget https://aws-codedeploy-us-east-2.s3.us-east-2.amazonaws.com/latest/install
chmod +x ./install; ./install auto
```

> Note: Change the source of the installer to match the AWS region your EC2 instance was created in. For example, if your AWS EC2 instance is in the us-west-1 AWS region, you'd download the installer from https://aws-codedeploy-us-west-1.s3.us-west-1.amazonaws.com/latest/install

- [ ] Verify the installation was successful by running the following command:

```
service codedeploy-agent status
```

## Java

> Note: Installing Java is optional, especially if you update the JRE when installing and configuring ColdFusion. However, if you update the ColdFusion Server using the java-jar method, this becomes mandatory.

- [ ] Run the following command:

```
apt install -y default-jre
```

- [ ] To confirm, run the following command:

```
java --version
```

## Zip

Since the ColdFusion installer is zipped, we will need unzip software on the server.

- [ ] Run the following command:

```
apt install -y zip
```

## ColdFusion 2023

- [ ] Run the following commands:

```
cd /home/ubuntu
wget https://s3.us-east-2.amazonaws.com/acf.cfsummit.vegas/CF2023_linux.zip
unzip CF2023_linux.zip
unzip ColdFusion_WWEJ_linux64.zip -d /opt/
mv /opt/ColdFusion/ /opt/coldfusion2023
cd /opt/coldfusion2023/cfusion/bin
./cfinstall.sh
```

- [ ] Follow the command prompts to configure ColdFusion Server.
    - [ ] Accept the license agreement
    - [ ] Select 30-Day Trial.

- [ ] Establish an Administrator password.
- [ ] Establish the built in web server port number.
- [ ] Establish an RDS password.
- [ ] Select the default user.
- [ ] Specify Development as the type of deployment.
- [ ] Proceed.

- [ ] Once ColdFusion has been configured, clean up the installer files by running the following commands:

```
cd /home/ubuntu
rm -rf CF2023_linux.zip
rm -rf ColdFusion_WWEJ_linux64.zip
rm -rf META-INF
rm -rf install
rm -rf install-aws-code-deploy
```

- [ ] Finally, start the ColdFusion server by running the following command:

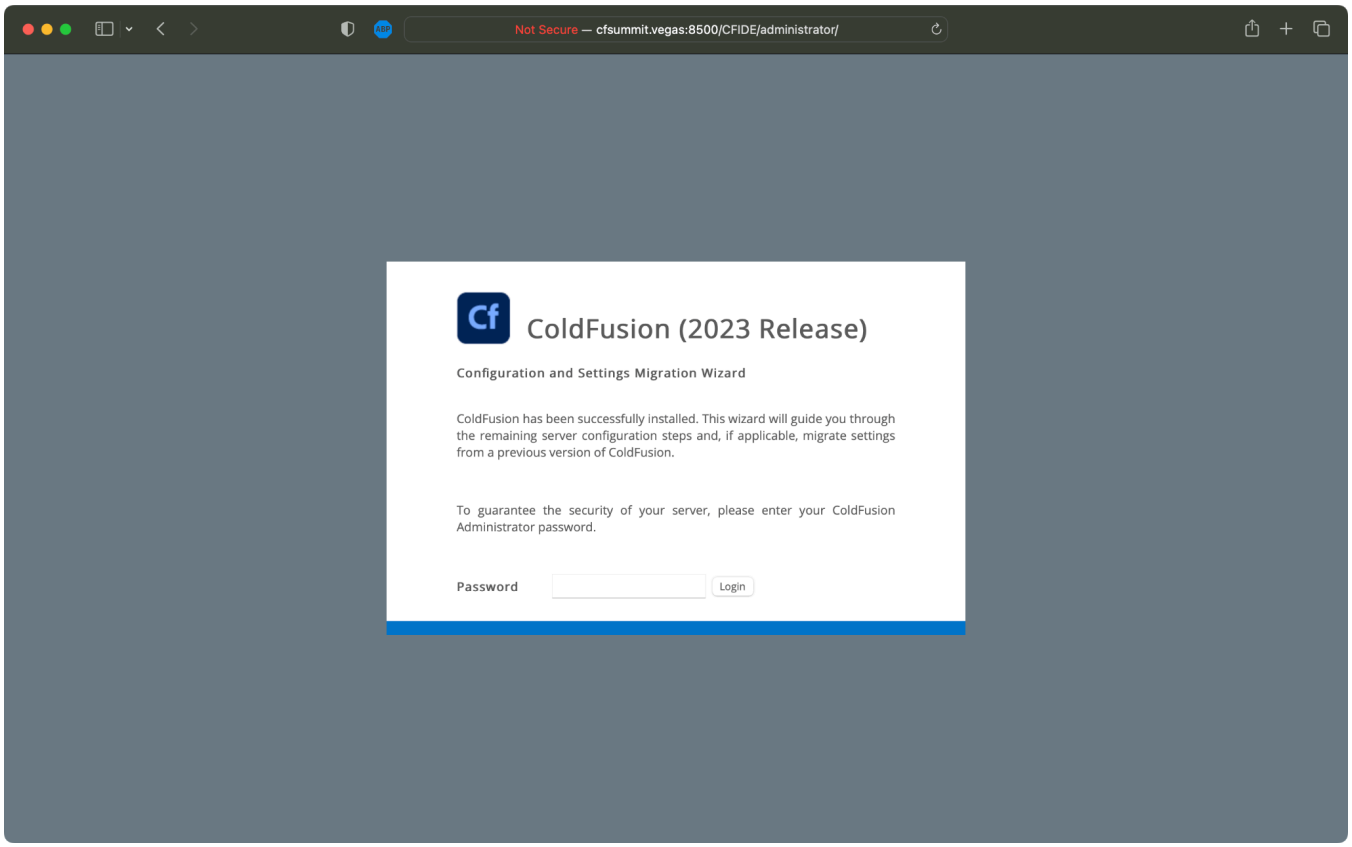
```
cd /opt/coldfusion2023/cfusion/bin
./coldfusion start
```

- [ ] Reboot the server by running the following command:

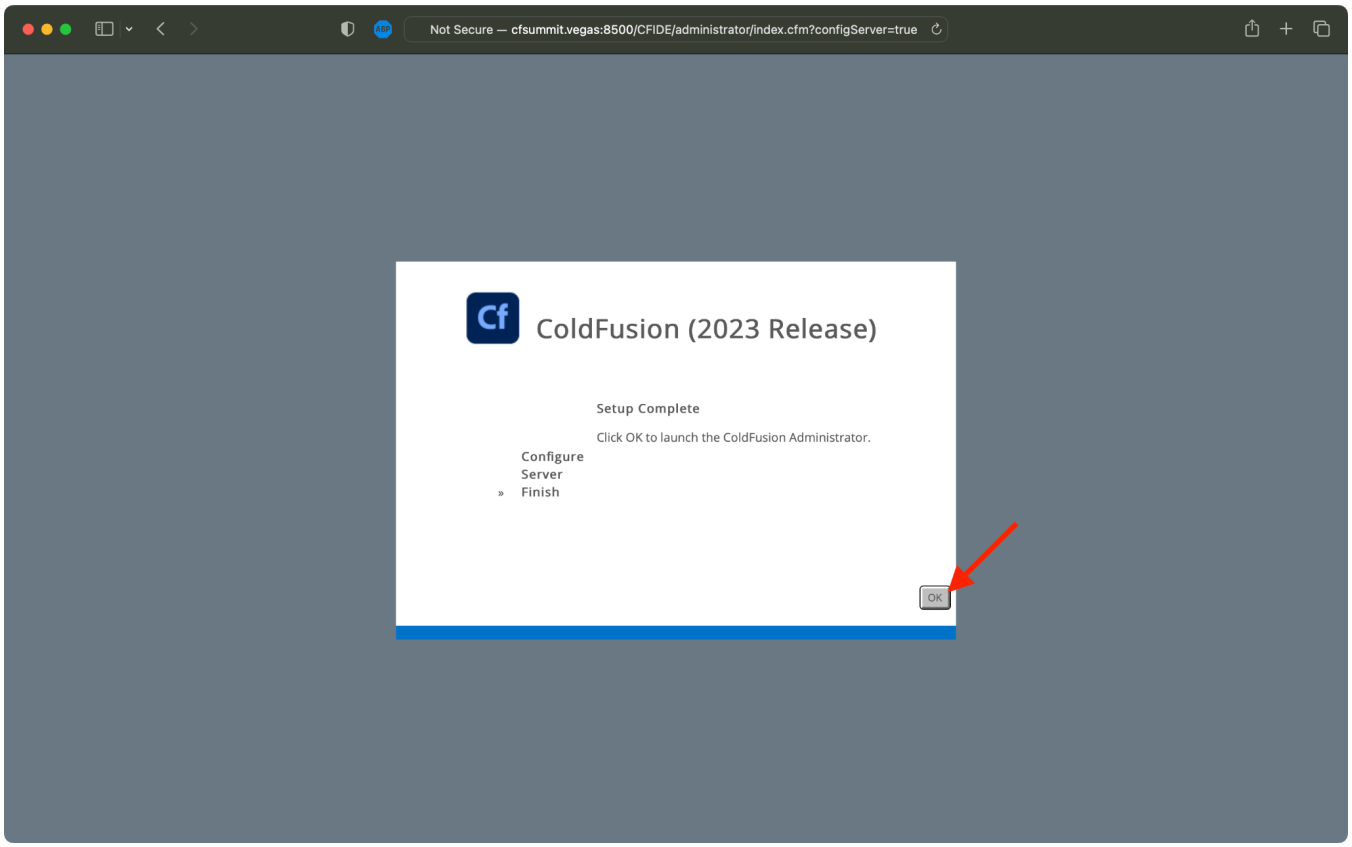
```
reboot now
```

## Log into the ColdFusion Administrator

- [ ] Browse to the ColdFusion Administrator at **`http://cfsummit.vegas:8500/CFIDE/administrator`**.
- [ ] The Configuration and Settings Migration Wizard should appear.

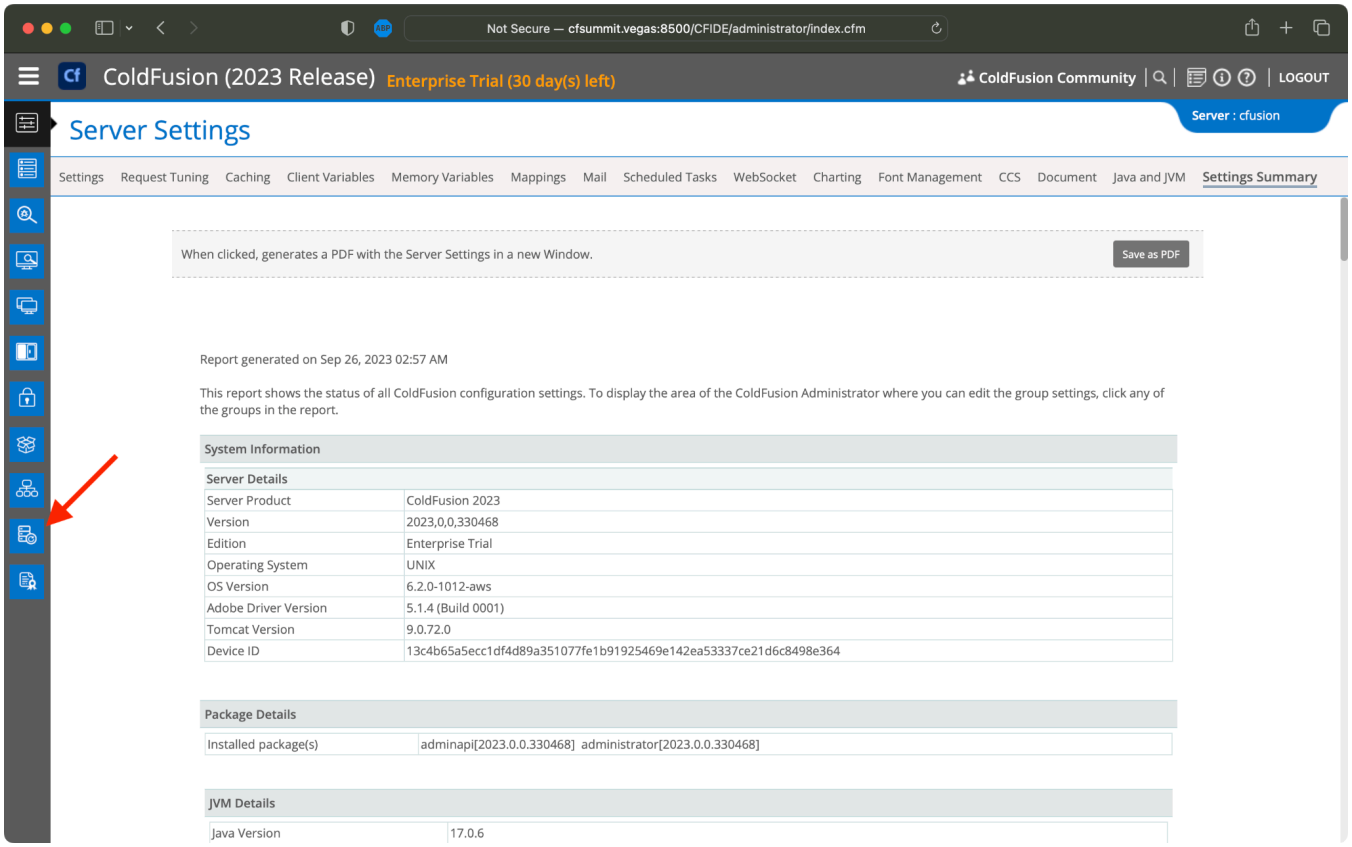



- ☐ Enter the ColdFusion Administrator Password created during setup.
- ☐ Once the server configuration setup is complete, click the OK button.

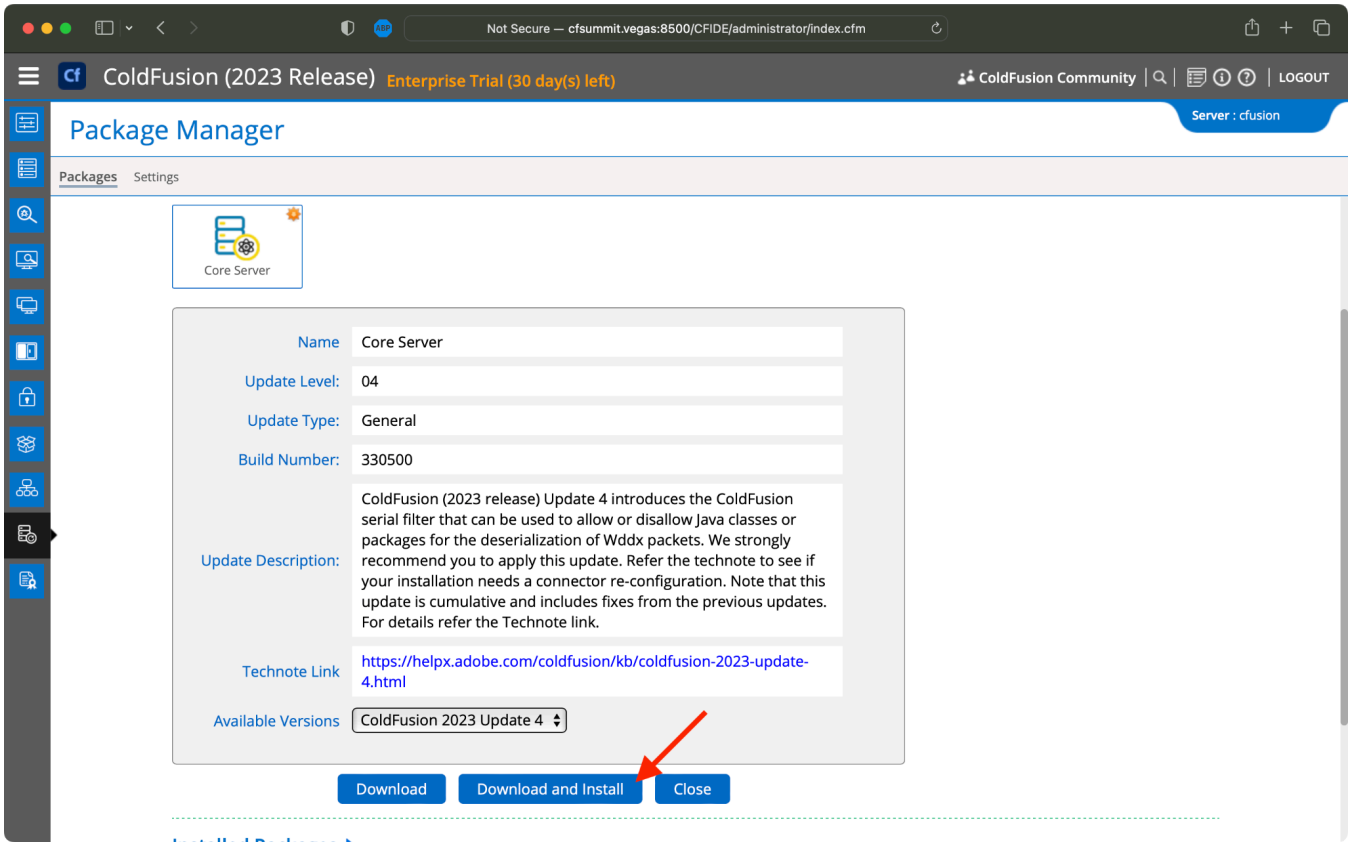


# Install ColdFusion Updates via the ColdFusion Administrator

- [ ] Once logged in, browse to the Package Manager.




- [ ] Expand the Core Server section.
- [ ] Click Download and Install to install the latest ColdFusion 2023 hotfix.




☐ Follow the prompts until the installation is complete.

## Alternatively, Download and Install ColdFusion 2023 Updates using Java

> Note: The Oracle Java CPU update released in July 2023 introduced an additional validation of ZIP64 extra fields that is known to cause an issue while downloading and/or applying the ColdFusion updates that were released on July 19, 2023 and Aug 17, 2023. The "-Djdk.util.zip.disableZip64ExtraFieldValidation=true" directive below allows the hotfix to be applied without issue.

☐ Run the following commands:

```
cd /opt/coldfusion2023/cfusion/bin
./coldfusion stop
cd /home/ubuntu
wget
https://cfdownload.adobe.com/pub/adobe/coldfusion/2023/updates/hotfix-006-330617
.jar
java -Djdk.util.zip.disableZip64ExtraFieldValidation=true -jar
hotfix-006-330617.jar
```

☐ Accept the terms of the licensing agreement.

- [ ] Once the hotfix has been installed, clean up the hotfix files by running the following command:

```
rm -rf /home/ubuntu/hotfix-004-330500.jar
```

### Configure Apache using the Web Server Connector

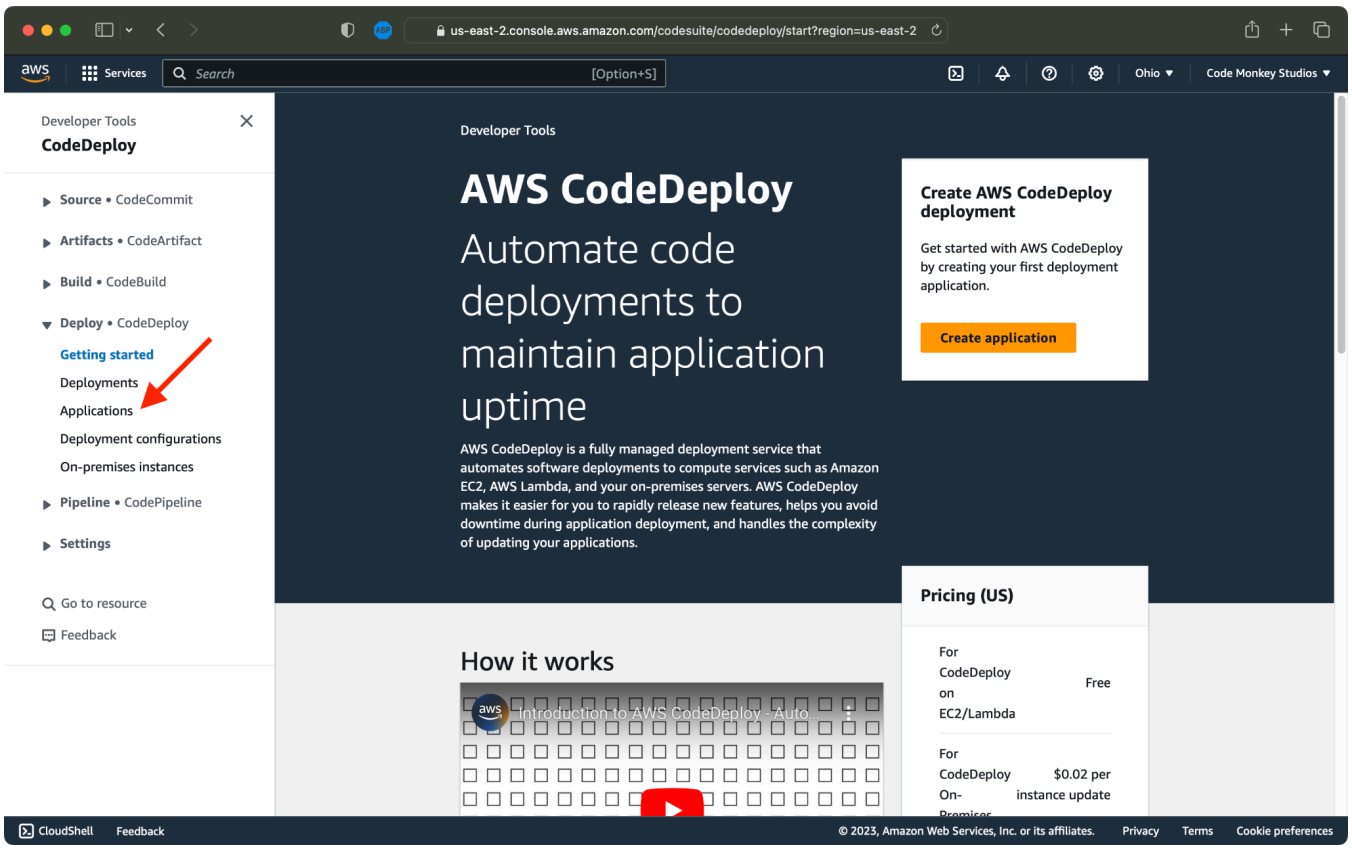
- [ ] Run the following commands:

```
cd /opt/coldfusion2023/cfusion/runtime/bin/
./wsconfig -ws Apache -dir /etc/apache2 -v
```

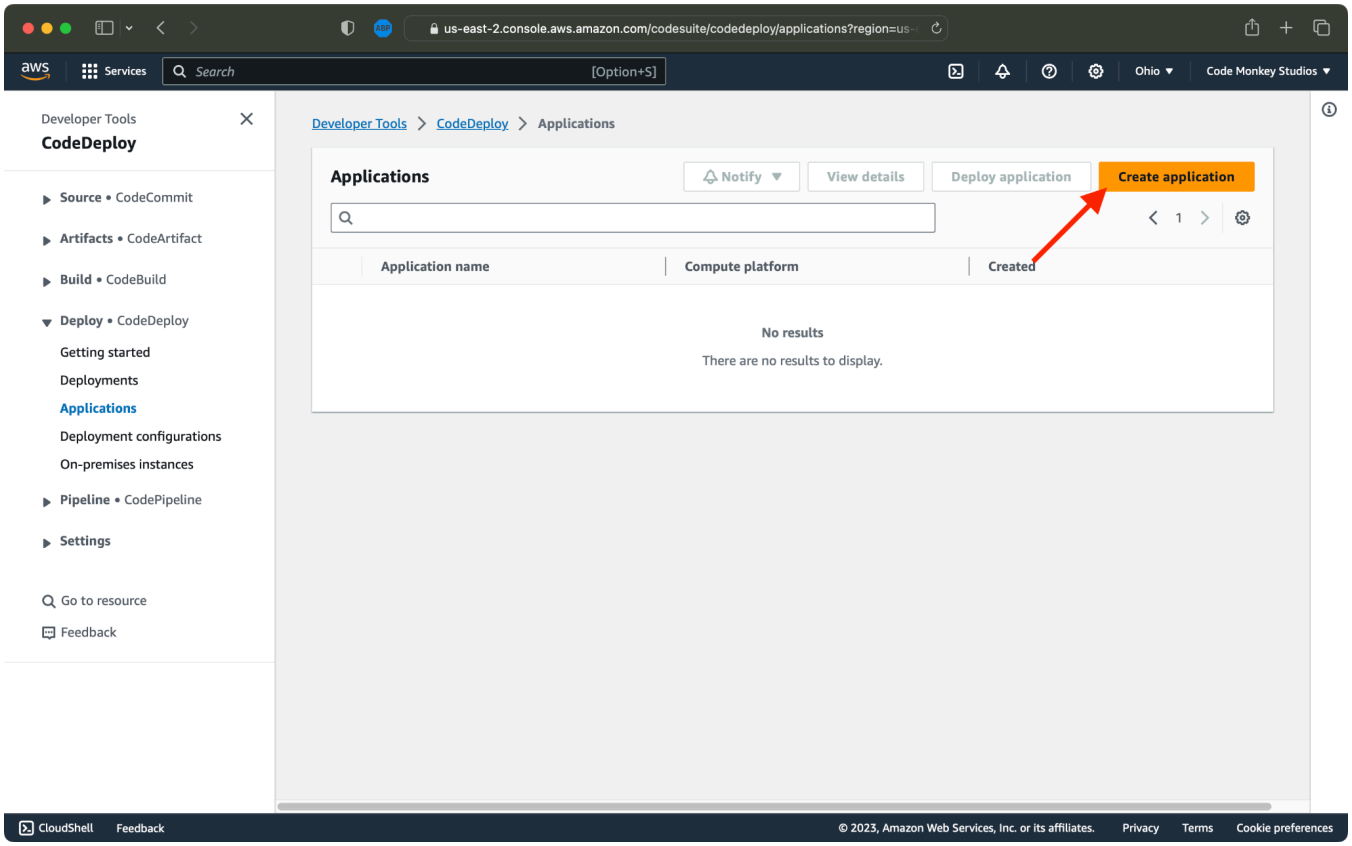
# Part 5 - AWS CodeDeploy Configuration

## Step 1 - Create an Application

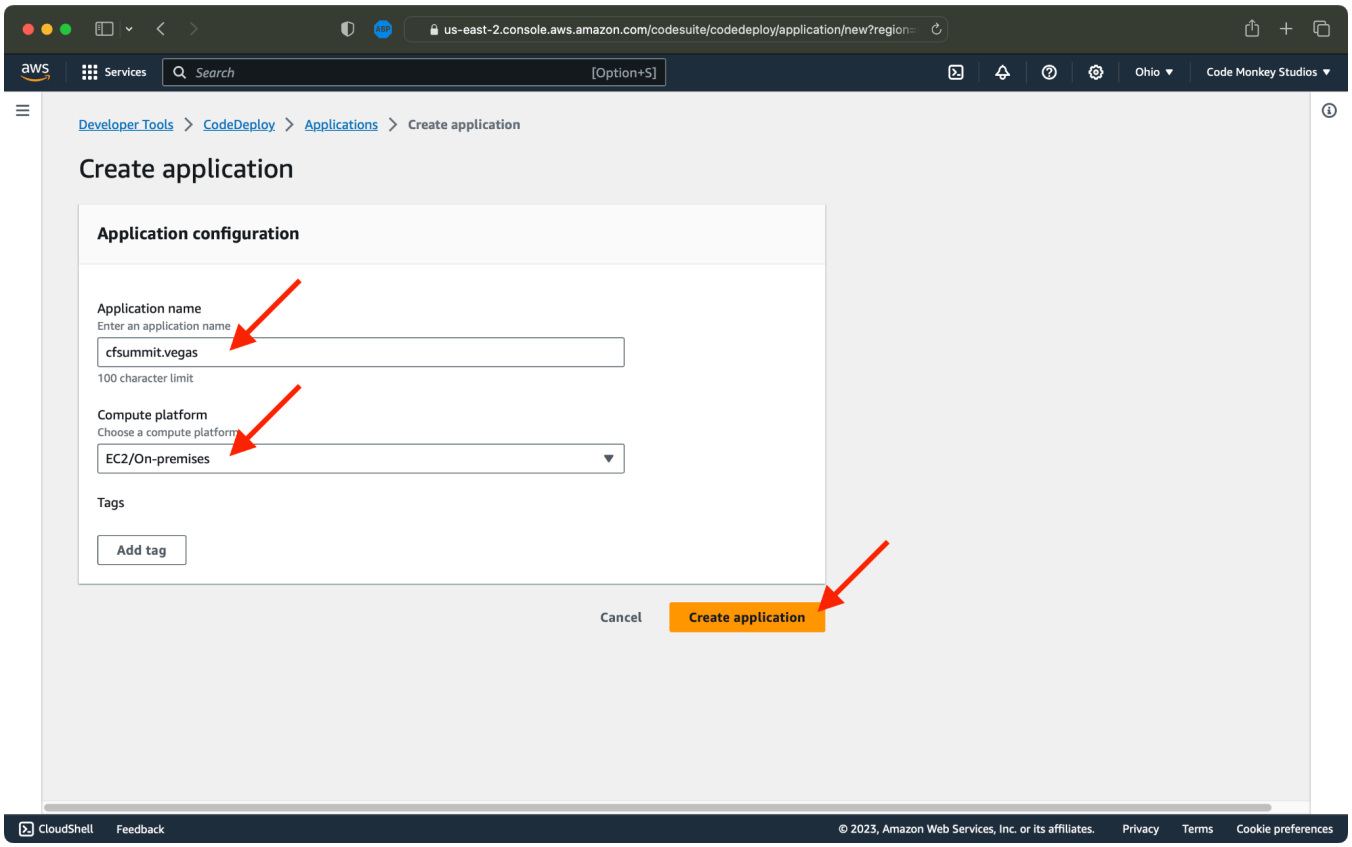
- [ ] Go to the AWS CodeDeploy service.
- [ ] Click on the Application link in the left column.




- [ ] Click the Create Application button.

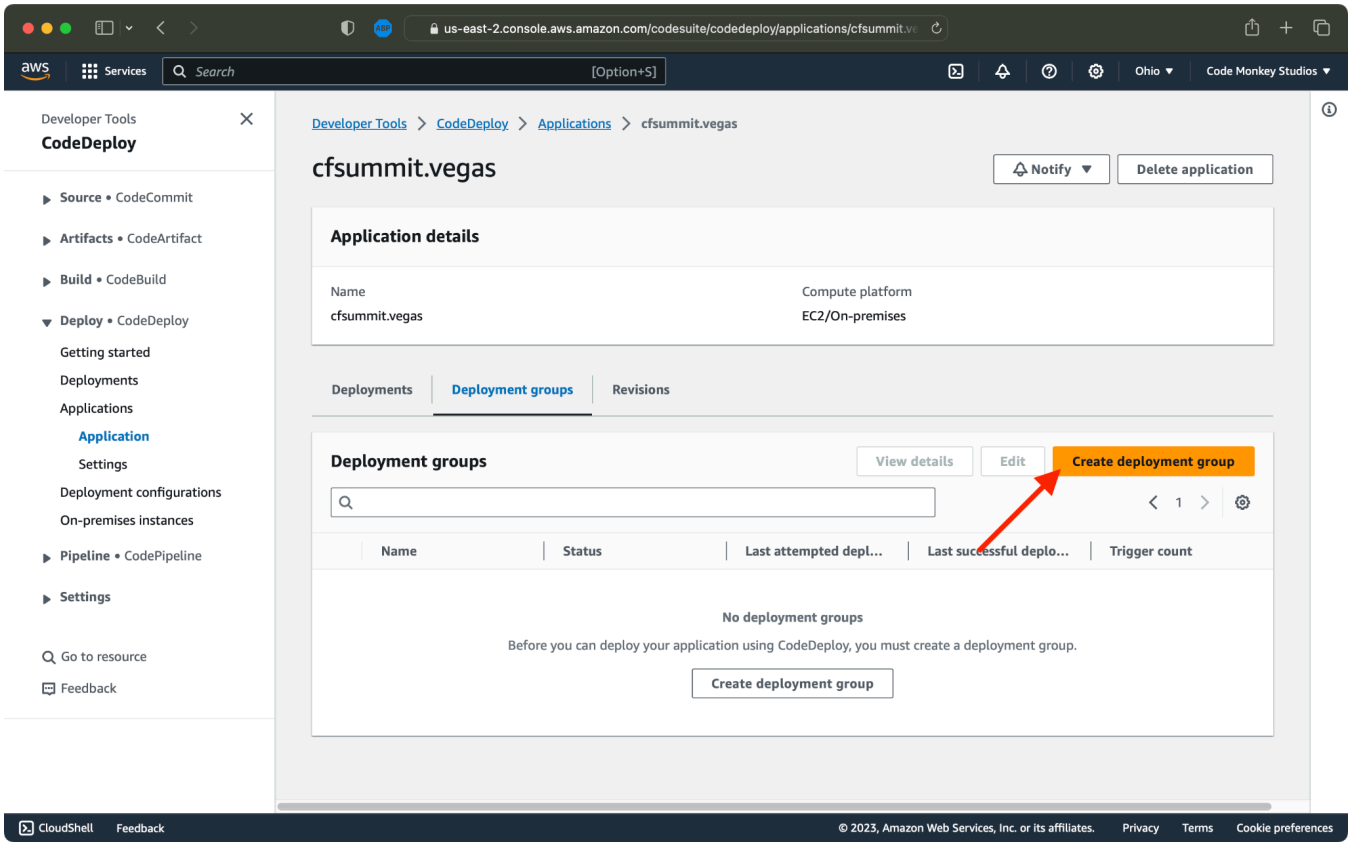



- [ ] Name the application `cfsummit-vegas-production`.
- [ ] Select EC2/On Premises from the Compute Platform dropdown menu.
- [ ] Click the Create Application button.



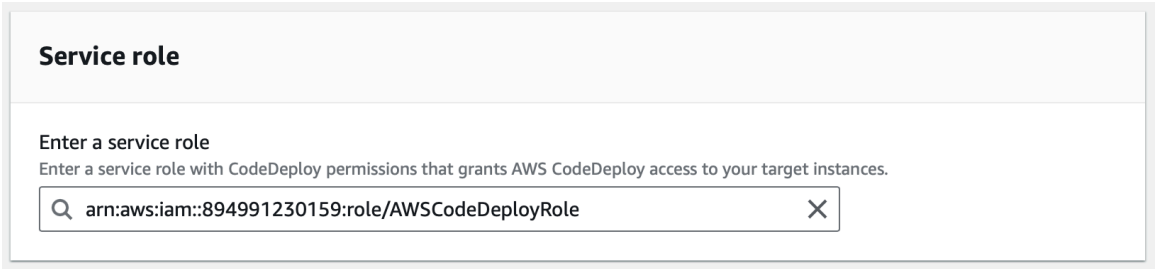

## Step 2 - Create a Deployment Group

- [ ] Click on the Create Deployment Group button.

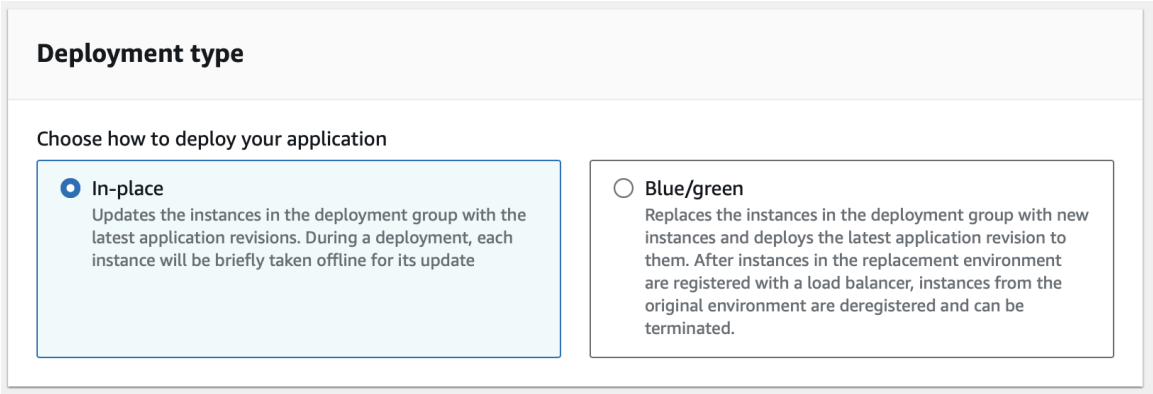



☐ In the section titled Deployment Group Name, name the deployment group `cfsummit-vegas-production-deployment-group`.

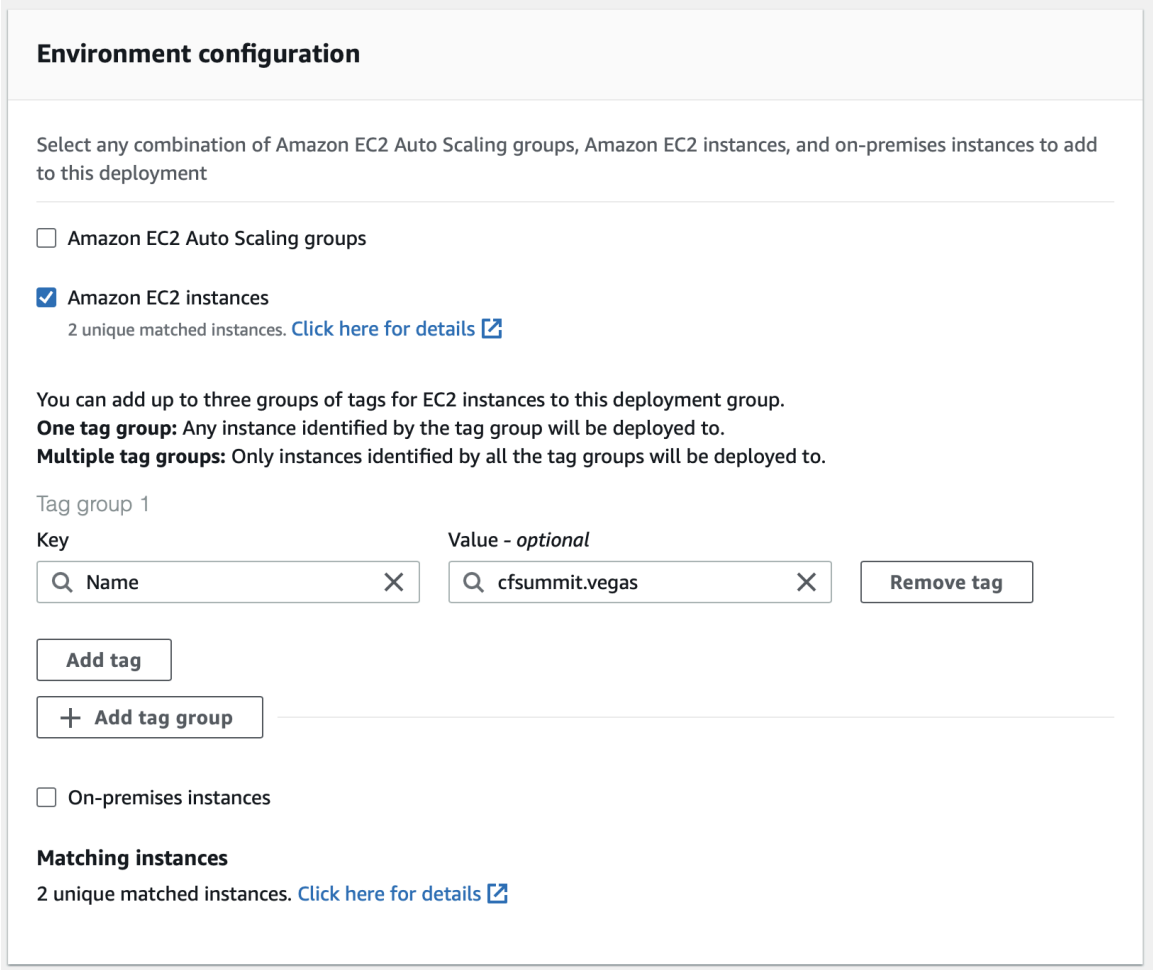

☐ In the section titled Service Role, select the AWSCodeDeployRole service role from the Service Role dropdown.



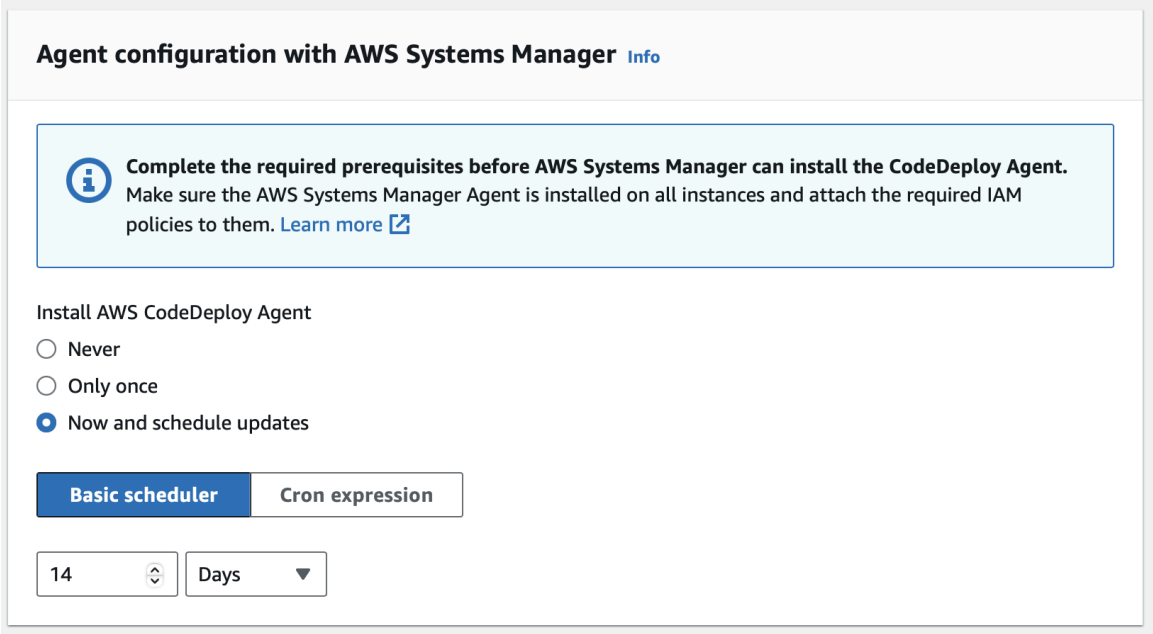

☐ In the section titled Deployment Type, select the In-Place radio button.



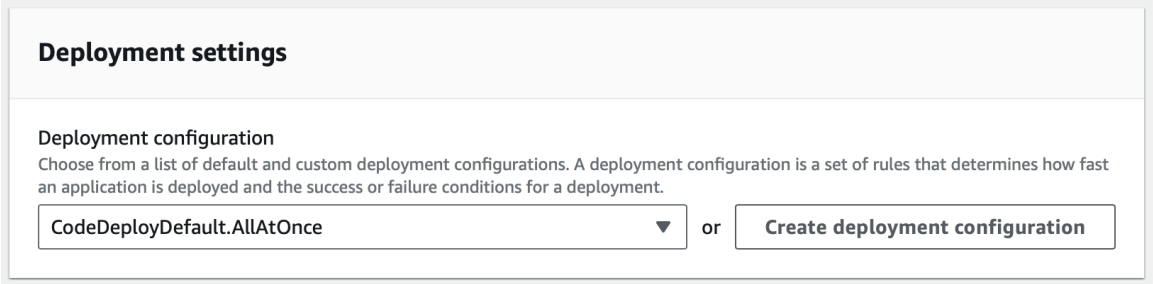

- [ ] In the section titled Environment Configuration:
  - [ ] Select the Amazon EC2 checkbox.
  - [ ] Select the tag Name.
  - [ ] Select the value cfsummit.vegas.



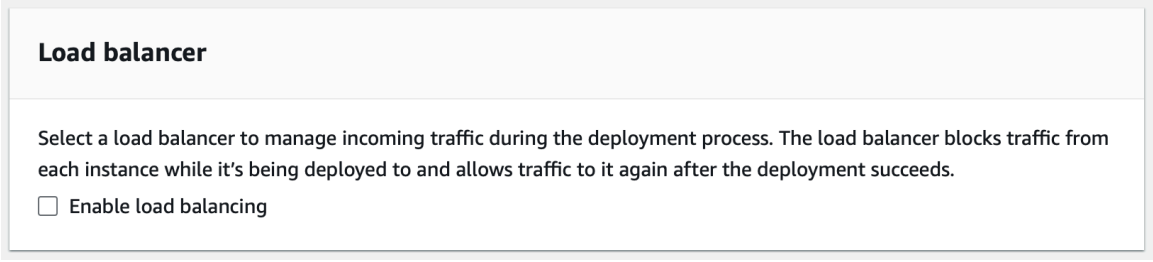

- [ ] In the section titled Agent Configuration with AWS Systems Manager, leave the default values.




☐ In the section titled Deployment Settings, leave the default value. (CodeDeployDefault.AllAtOnce)




☐ In the section labeled Load Balancer, uncheck the Enable Load Balancing checkbox.




☐ Click the Create Deployment Group button at the bottom of the page.

# Part 6 - BitBucket Pipeline Configuration

## Step 1 - Create the appspec.yml and bitbucket-pipelines.yml files

- ☐ In the root directory of the code repository, create an `appspec.yml` file with the following contents:

```
version: 0.0
os: linux
files:
  - source: app/
    destination: /var/www/cfsummit.vegas
```

- ☐ Create a `bitbucket-pipelines.yml` file with the following contents:

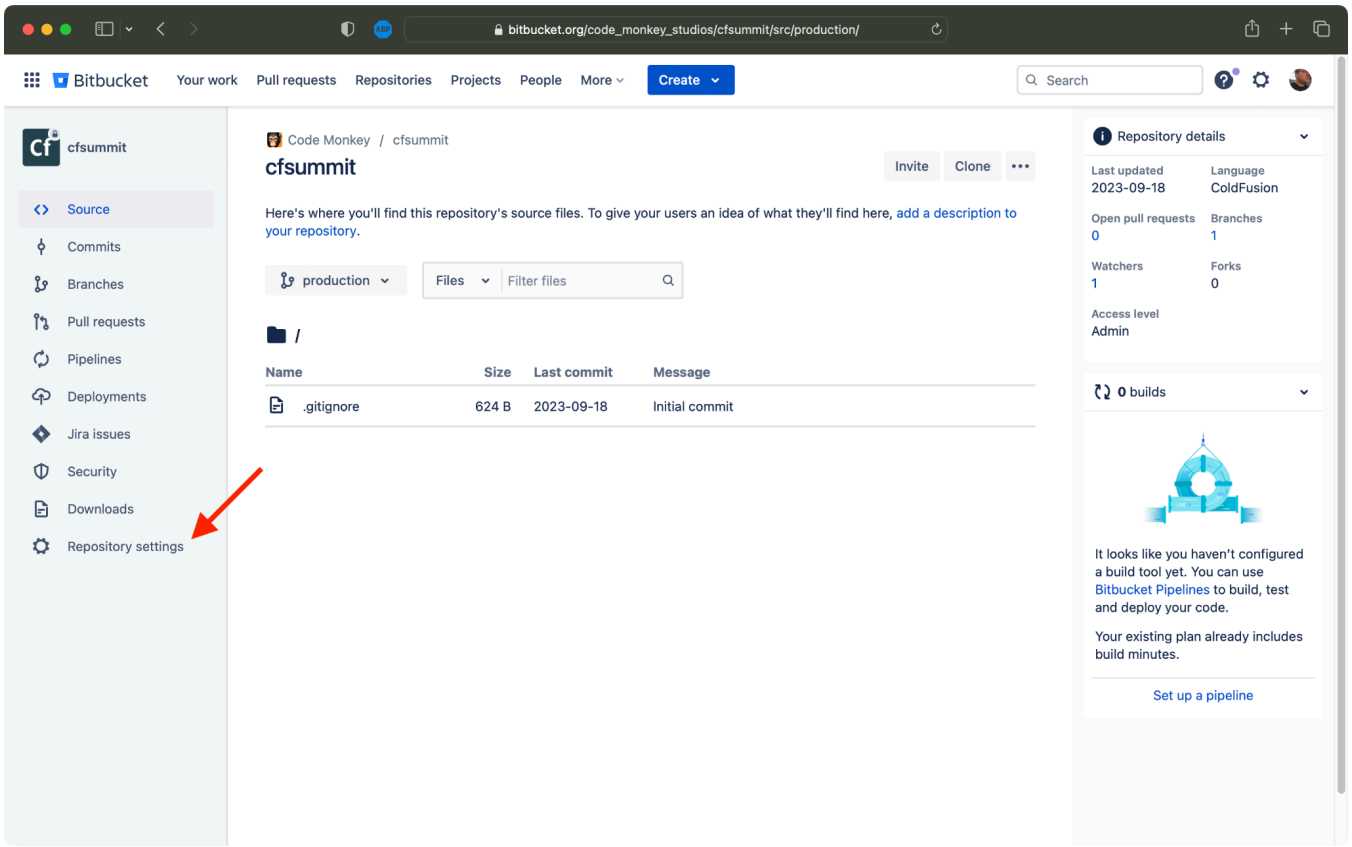
```
image: atlassian/default-image:3

pipelines:
  branches:
    production:
      - step:
          name: Deploy to cfsummit.vegas Production
          deployment: production
          script:
            - apt-get update
            - apt-get install -y zip
            - zip -r cfsummit-vegas-production.zip .
            - pipe: atlassian/aws-code-deploy:1.4.0
              variables:
                AWS_ACCESS_KEY_ID: $AWS_ACCESS_KEY_ID
                AWS_SECRET_ACCESS_KEY: $AWS_SECRET_ACCESS_KEY
                AWS_DEFAULT_REGION: $AWS_DEFAULT_REGION
                COMMAND: 'upload'
                APPLICATION_NAME: 'cfsummit-vegas-production'
                BUNDLE_TYPE: 'zip'
                ZIP_FILE: 'cfsummit-vegas-production.zip'
            - pipe: atlassian/aws-code-deploy:1.4.0
              variables:
                AWS_ACCESS_KEY_ID: $AWS_ACCESS_KEY_ID
                AWS_SECRET_ACCESS_KEY: $AWS_SECRET_ACCESS_KEY
                AWS_DEFAULT_REGION: $AWS_DEFAULT_REGION
                COMMAND: 'deploy'
```

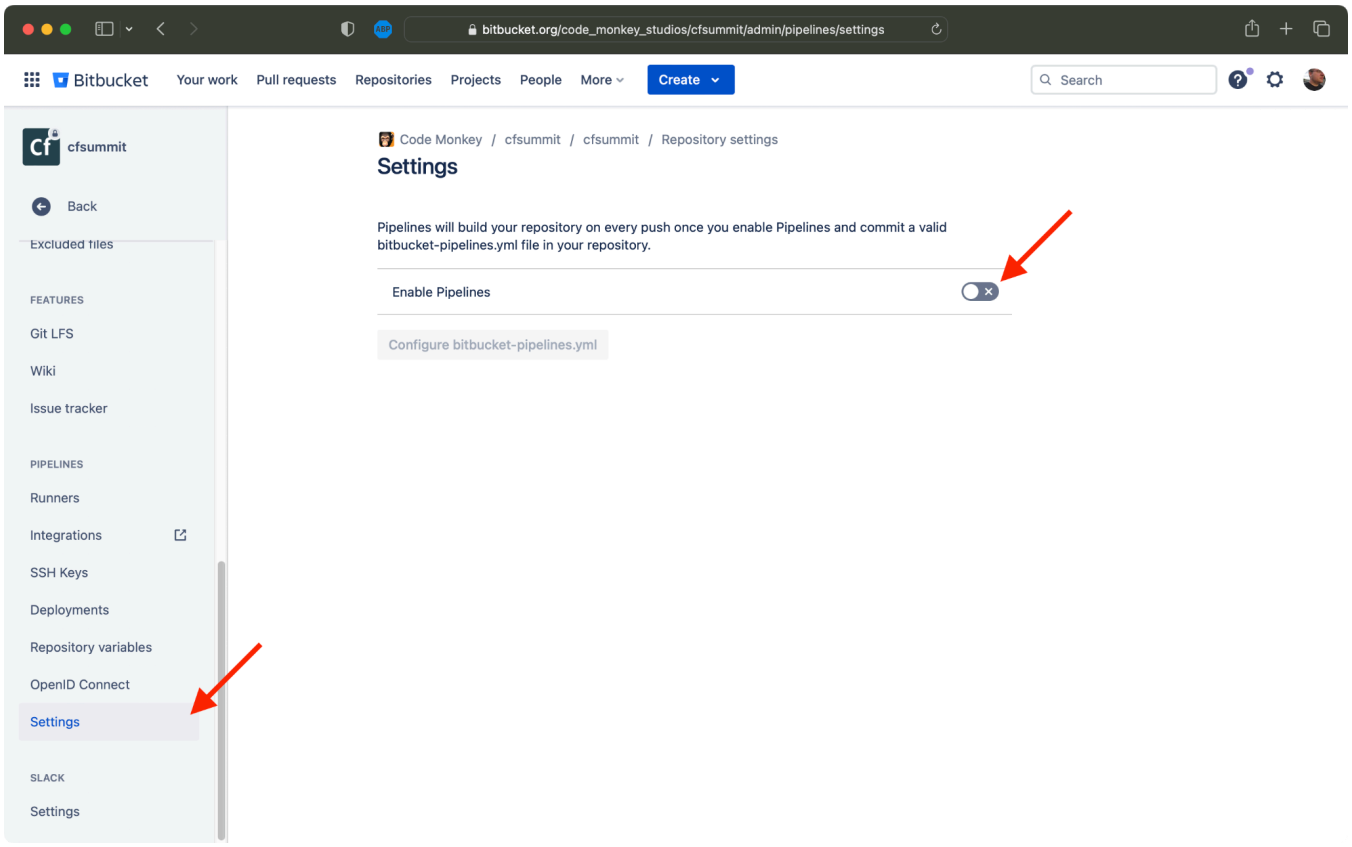
```
            APPLICATION_NAME: 'cfsummit-vegas-production'
            DEPLOYMENT_GROUP: 'cfsummit-vegas-production-deployment-group'
            WAIT: 'true'
            FILE_EXISTS_BEHAVIOR: 'OVERWRITE'
            IGNORE_APPLICATION_STOP_FAILURES: 'true'
```

## Step 2 - Create the Repository Variables in BitBucket

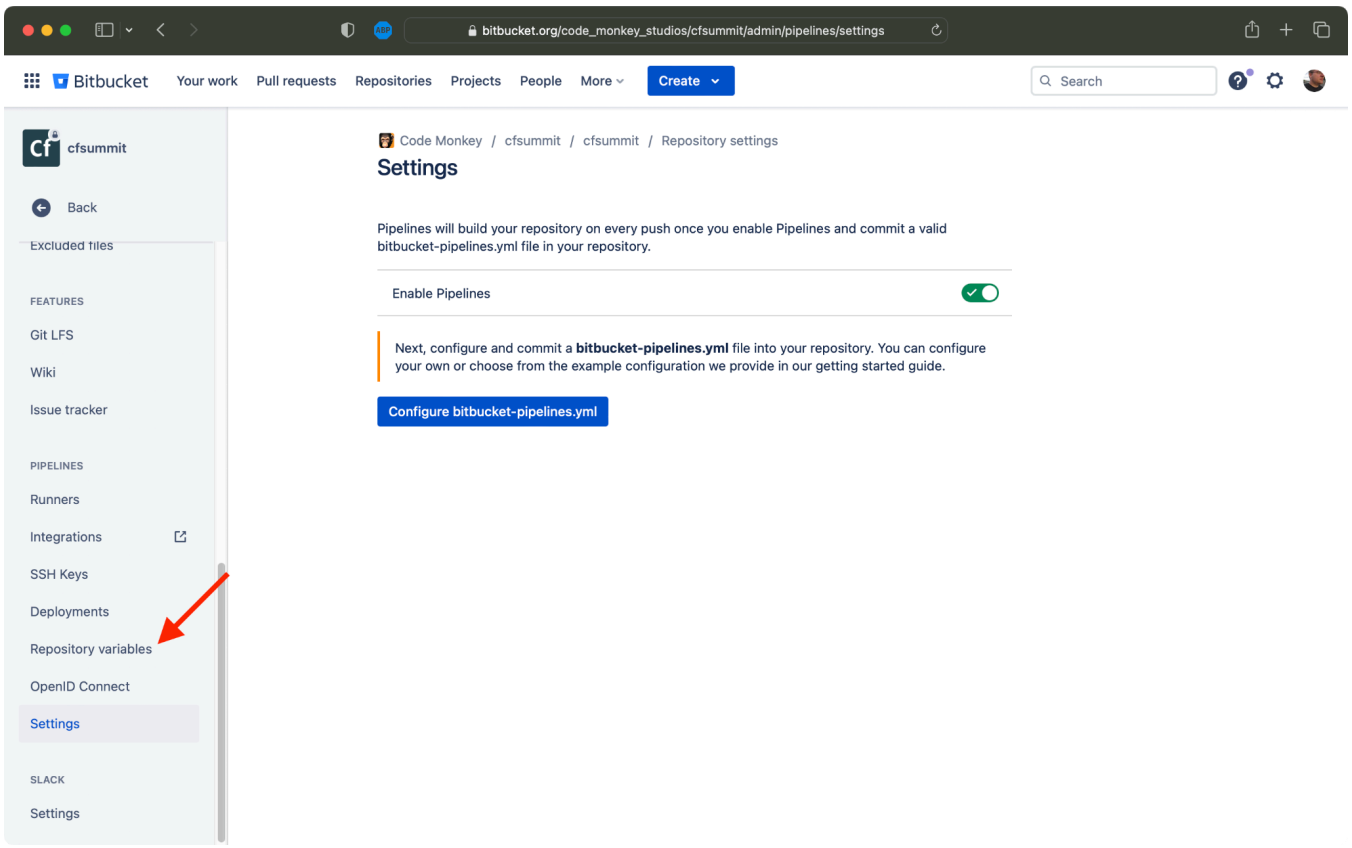
- [ ] From our repository in BitBucket, click the Repository Settings link in the left column.



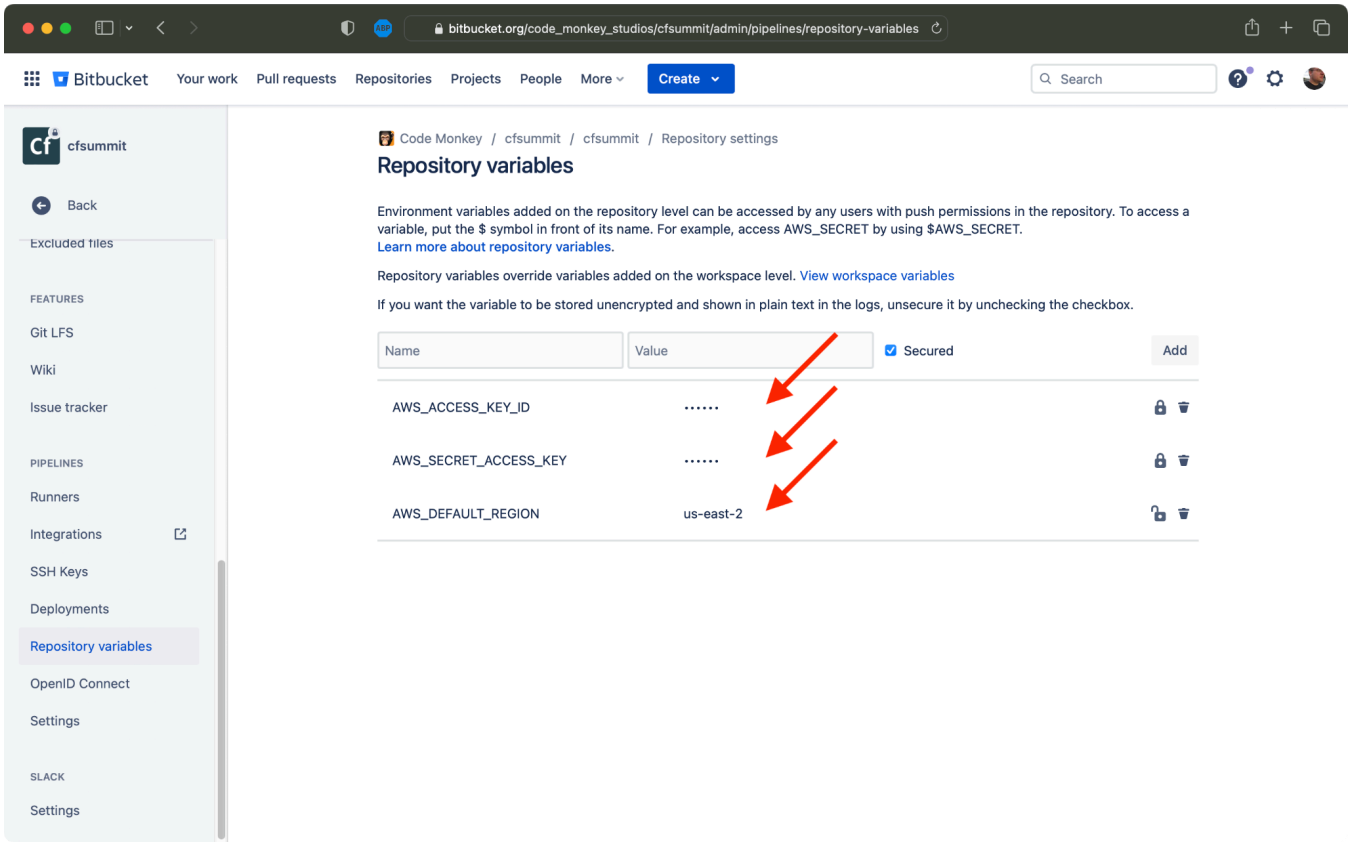

- [ ] Scroll to the bottom of the left column and click on the Settings link.
- [ ] Toggle the Enable Pipelines toggle.




☐ Next, click the Repository Variables link in the left column.

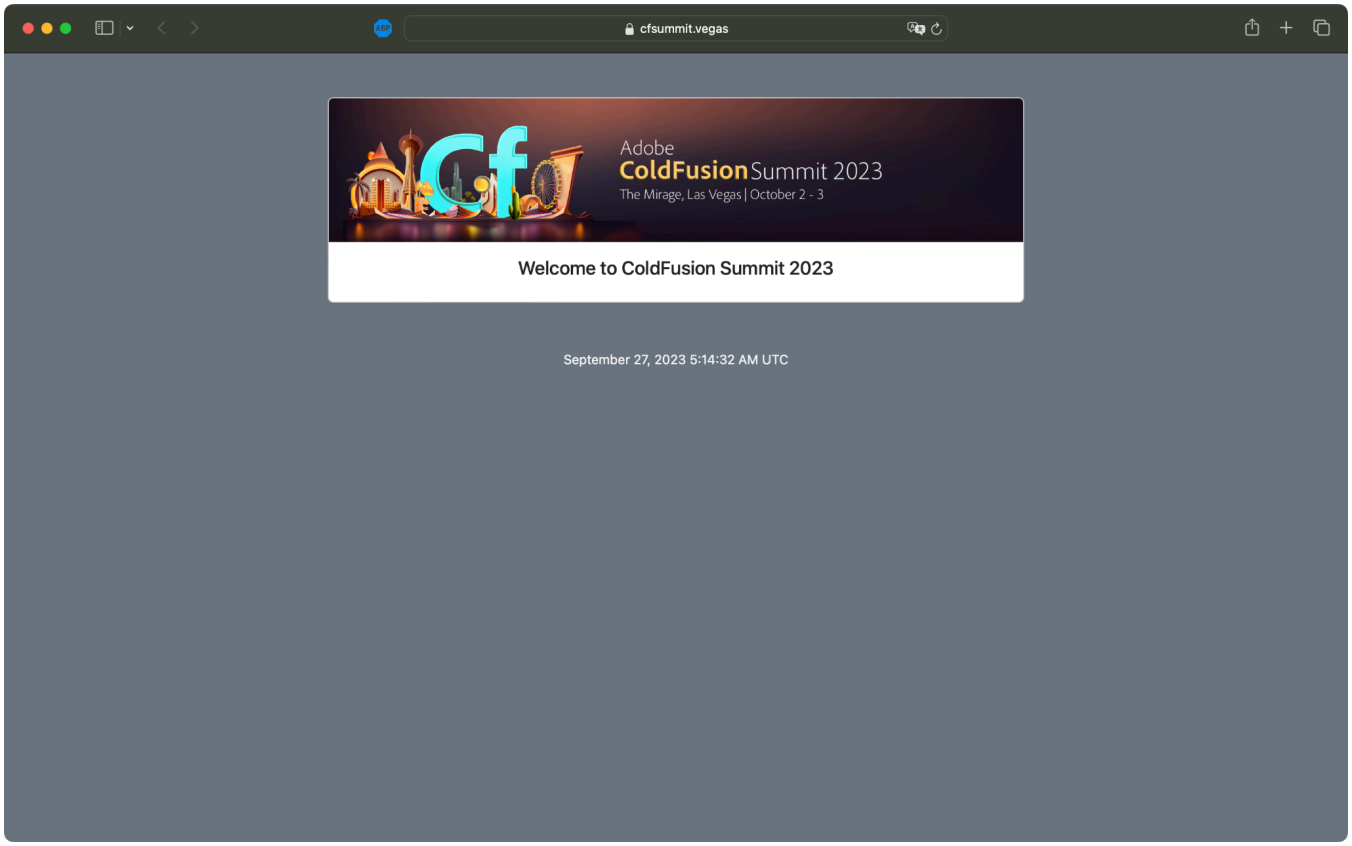


- [ ] From the Repository Variables page, create four variables:
    - [ ] Non-secure variable:
        - [ ] AWS_DEFAULT_REGION, with a value of us-east-2.
    - [ ] Secure variables:
        - [ ] AWS_ACCESS_KEY_ID, with a value copied from the .csv credentials file we downloaded earlier.
        - [ ] AWS_SECRET_ACCESS_KEY, with a value copied from the .csv credentials file we downloaded earlier.




# Part 7 - Moment of Truth

If we've done everything right, we should see a successful deployment, and our code will be pushed live to the server.


cfsummit.vegas
Adobe
ColdFusion Summit 2023
The Mirage, Las Vegas | October 2 - 3
Welcome to ColdFusion Summit 2023
September 27, 2023 5:14:32 AM UTC