😊 Creating a Pipeline between S3 buckets

CI/CD stands for **Continuous Integration** and **Continuous Delivery** (or **Continuous Deployment**). It's a set of practices used in software development to make the process of getting code from development to production smoother and more reliable. Here's a simple breakdown:

Continuous Integration (CI)

1. **Code Integration**: Developers frequently merge their code changes into a shared repository, often several times a day.
2. **Automated Testing**: Each time code is integrated, automated tests run to check if the new changes work well with the existing code. This helps catch bugs early.

In simple terms: Imagine you're working on a group project. Every time you finish a part, you add it to the main project folder. Someone checks if it fits well with everyone else's work, and if it doesn't, you fix it right away. This way, you always know the project is in good shape.

Continuous Delivery (CD)

1. **Automated Release Process**: After passing tests in CI, the code is automatically prepared for release. This can involve packaging the code and ensuring everything is ready for deployment.
2. **Manual Approval (sometimes)**: Before the new version of the software goes live, there may be a manual approval step where someone reviews the changes.

In simple terms: Once your part of the project is checked and fits well, it's packaged up nicely and put in a place where it can be easily accessed. Someone might give it a final check before it's officially shared with everyone.

Continuous Deployment (CD)

1. **Automatic Deployment**: In this case, after the code passes all tests and checks, it's automatically deployed to production without manual approval.

In simple terms: Instead of waiting for a final check, the project update is automatically shared with everyone as soon as it's ready.

Overall, CI/CD helps developers quickly and safely deliver new features, updates, and fixes to users. It ensures that the software is always in a working state and that changes can be released frequently with confidence.

1. CI/CD Pipeline

A CI/CD pipeline is a set of automated processes that software goes through from code changes to deployment. The key stages typically include:

- **Source Stage**: This is where the code is stored in a version control system (like Git). When new changes are pushed to the repository, it triggers the CI/CD pipeline.

- **Build Stage**: The code is compiled and built into an executable format. This step might also include packaging the code with its dependencies.
- **Test Stage**: Automated tests are run to validate the code. Tests can include unit tests (checking individual components), integration tests (checking how components work together), and end-to-end tests (checking the whole application).
- **Deploy Stage**: The code is deployed to a production or staging environment. In continuous deployment, this happens automatically; in continuous delivery, there might be a manual approval step.

2. Benefits of CI/CD

- **Faster Time to Market**: By automating the integration, testing, and deployment processes, CI/CD enables teams to release new features and fixes more quickly.
- **Higher Quality**: Automated testing ensures that code changes are thoroughly tested, reducing the likelihood of bugs reaching production.
- **Consistent Delivery**: The automated nature of CI/CD reduces human error and ensures that deployments are consistent and reliable.
- **Improved Collaboration**: Developers can focus on writing code and rely on the pipeline to handle the integration and deployment, fostering a more collaborative environment.

3. Challenges in Implementing CI/CD

- **Initial Setup and Cost**: Setting up a robust CI/CD pipeline requires an initial investment in tools, infrastructure, and training. This can be time-consuming and costly.
- **Maintenance**: The pipeline itself needs to be maintained. As the codebase and team evolve, the pipeline might need updates and optimizations.
- **Test Coverage**: The effectiveness of CI/CD depends heavily on the quality and coverage of automated tests. If tests are incomplete or poorly written, issues can still slip through.
- **Cultural Shift**: CI/CD requires a cultural change within the organization. Teams must embrace automation, frequent releases, and a mindset of continuous improvement.

4. Popular CI/CD Tools

- **Jenkins**: An open-source automation server widely used for building CI/CD pipelines.
- **GitLab CI/CD**: Integrated CI/CD capabilities within the GitLab platform.
- **CircleCI**: A cloud-based CI/CD service that automates the software development process.
- **Travis CI**: A CI/CD service often used for open-source projects.
- **GitHub Actions**: Integrated CI/CD workflows within GitHub.

5. Use Cases

- **Web Development**: CI/CD is commonly used in web development to frequently update web applications with new features and bug fixes.
- **Mobile App Development**: Automating the build, test, and release process for mobile apps ensures a consistent experience for users.
- **Microservices**: In architectures with many small, independent services, CI/CD helps manage the deployment and integration of each service.

6. Security Considerations

- **Secure Pipelines**: Ensure that the CI/CD pipeline itself is secure, with proper access controls and monitoring.
- **Vulnerability Scanning**: Integrate security checks, such as vulnerability scans, into the pipeline to identify potential security issues early.

In summary, CI/CD is a powerful practice that enhances the efficiency and reliability of software development and deployment. By automating repetitive tasks and ensuring rigorous testing, CI/CD helps teams deliver high-quality software quickly and consistently.

🤔 What are we doing in this Lab?

In this exercise, you are setting up a basic CI/CD pipeline using AWS services. The main goal is to automate the deployment of a static website hosted on Amazon S3. Here's a summary of what you're doing and the end goal:

1. **Creating S3 Buckets:**
    - One bucket serves as the **source repository** where you upload a zip file containing your website's content.
    - The other bucket is the **production environment**, where the website is hosted and made publicly accessible.
2. **Configuring Buckets:**
    - Enabling versioning on the source bucket to track changes and manage deployments.
    - Setting permissions and enabling static website hosting on the production bucket to make the website publicly viewable.
3. **Setting Up a Pipeline in AWS CodePipeline:**
    - Creating a pipeline that automatically triggers when new files are uploaded to the source bucket.
    - Configuring the pipeline to deploy the contents from the source bucket to the production bucket.

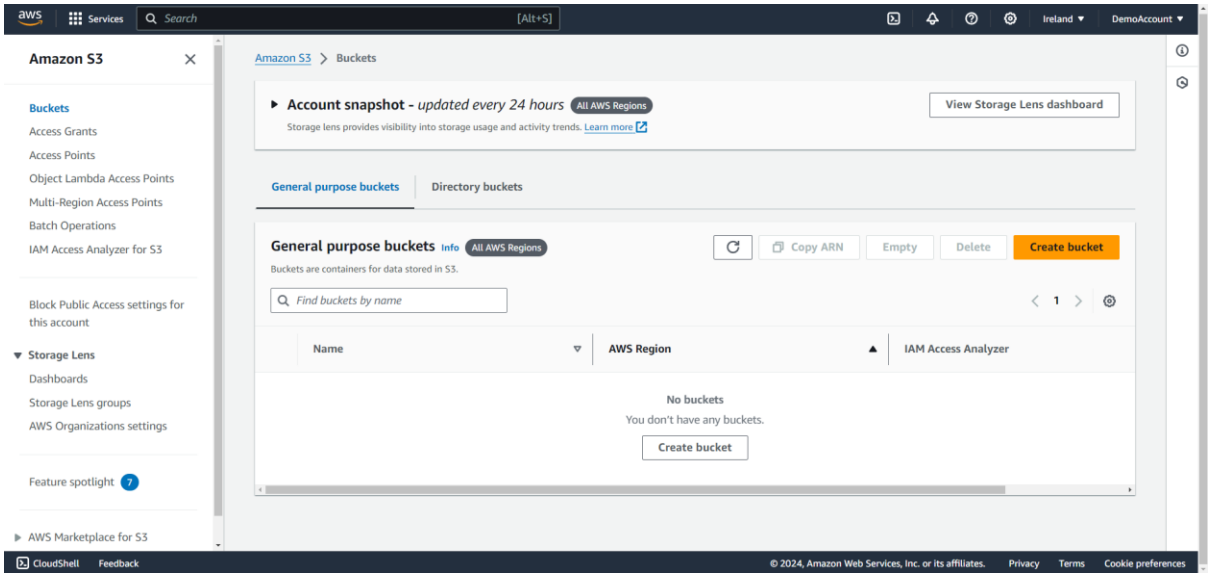
End Goal:

The objective is to automate the deployment process so that whenever you upload a new version of your static website to the source S3 bucket, the CodePipeline is triggered,

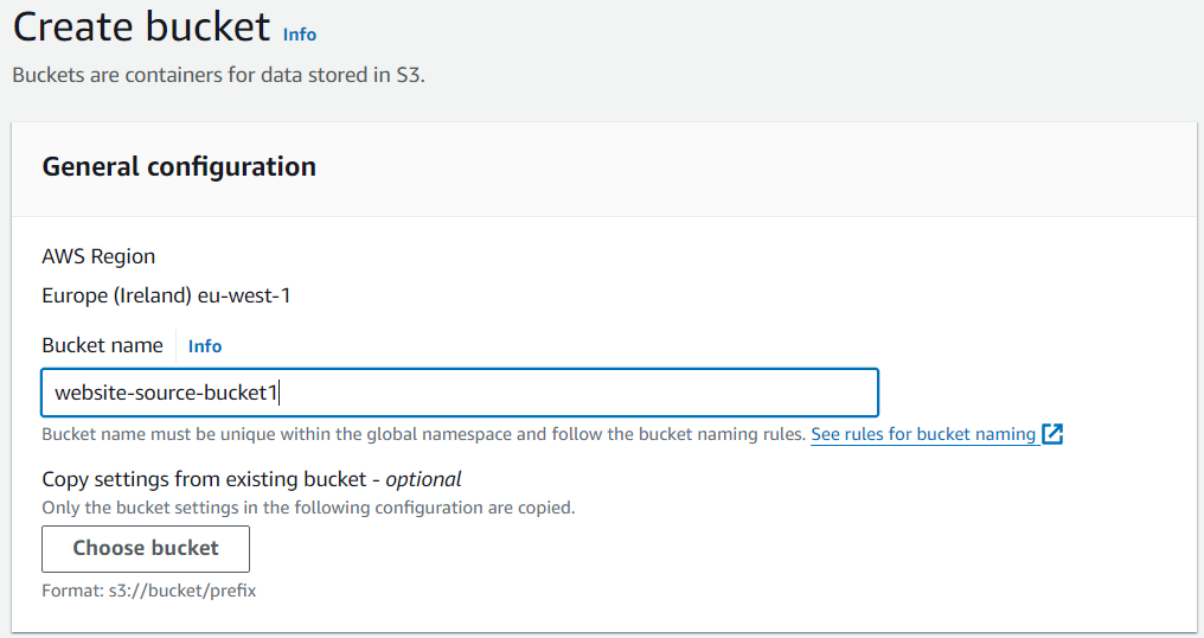
automatically deploying the new content to the production bucket. This setup ensures that your website is always up-to-date with the latest changes without manual intervention. The end result is a publicly accessible static website hosted on S3, with an automated process for updating the site content.

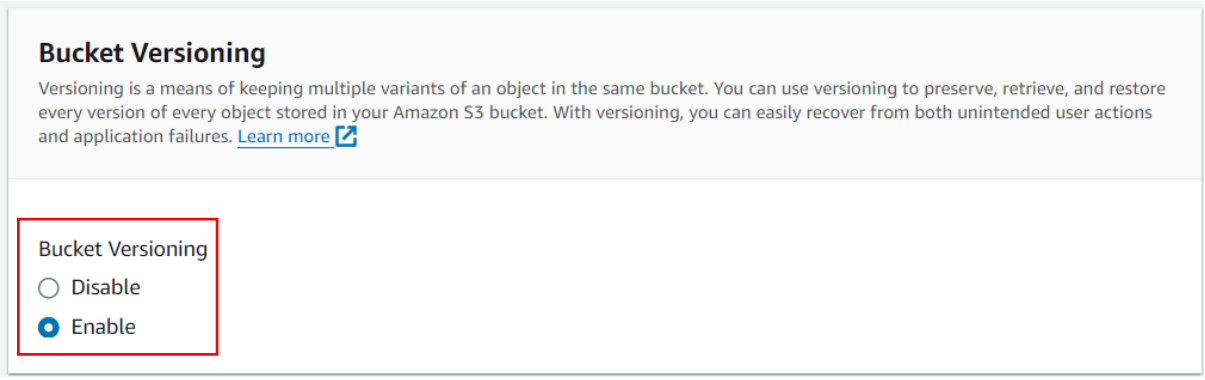
😄 To begin with the Lab:

1. In your AWS Console navigate to S3 and create two buckets. So, one bucket will be for the source repository and another bucket for the production environment to host a simple static website.
2. Then, we will build a pipeline on AWS Code Pipeline so that whenever we upload a new version of our website to the source bucket as a zip package, the pipeline will be triggered, and it will deploy the package contents to the production S3 bucket.
3. Now go to your AWS Console and then navigate to S3. Here we are going to create two buckets. So, click on Create Bucket.

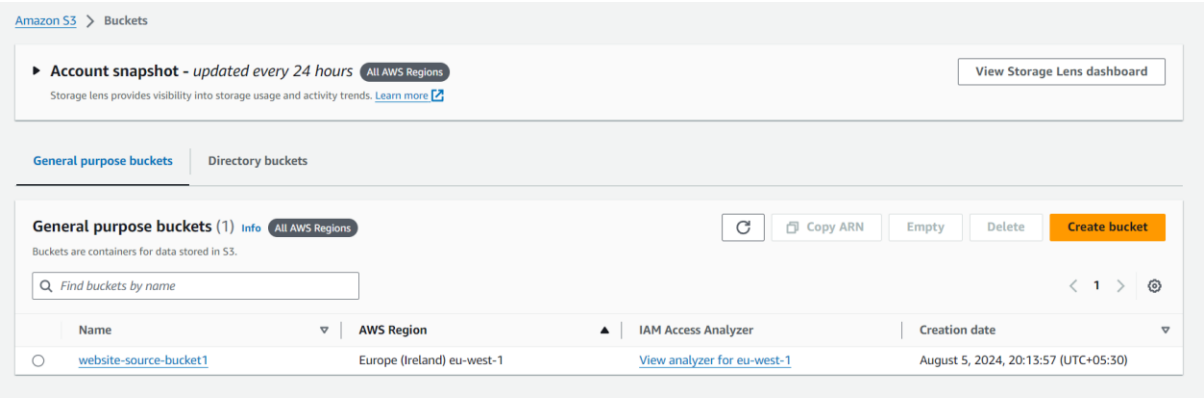


4. There are some crucial configurations you need to make for your bucket to be accepted as a source location by your pipeline. First, your source bucket must be in the same AWS region as your pipeline.
5. First, you need to give your bucket a name then scroll down.
6. Now, there is another requirement for S3 bucket source locations. You must enable versioning on your source bucket. This is because when you use an S3 bucket as the source location, Code Pipeline identifies source revisions from their S3 version IDs. After that just go ahead and create your bucket.

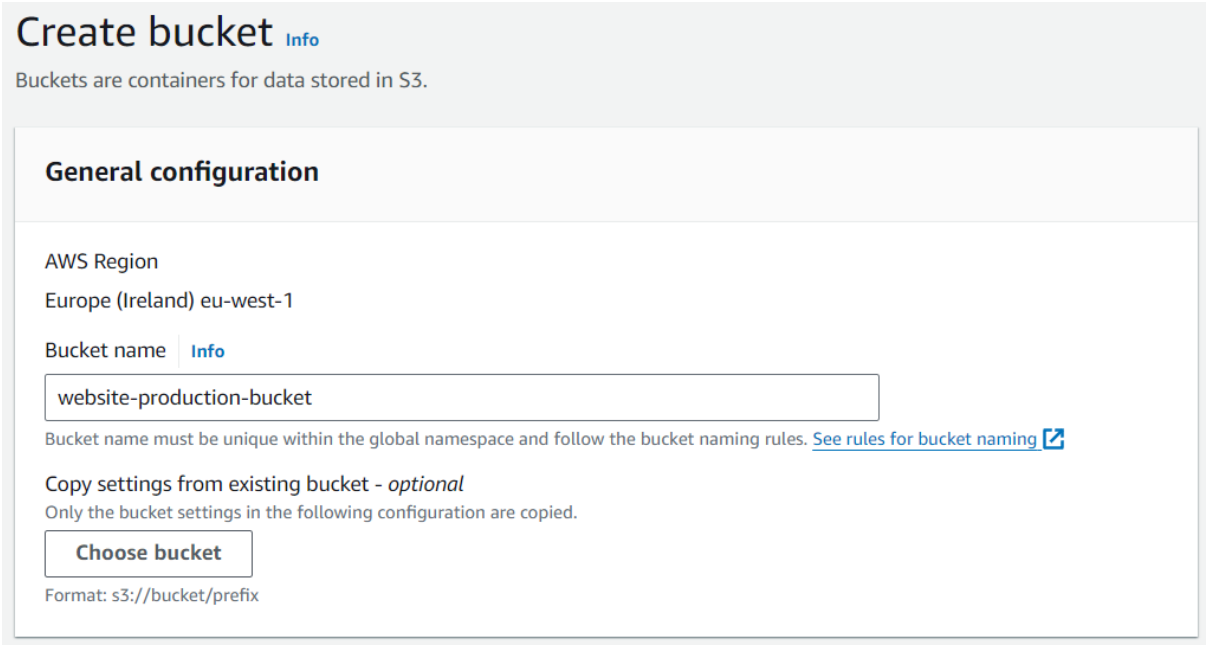




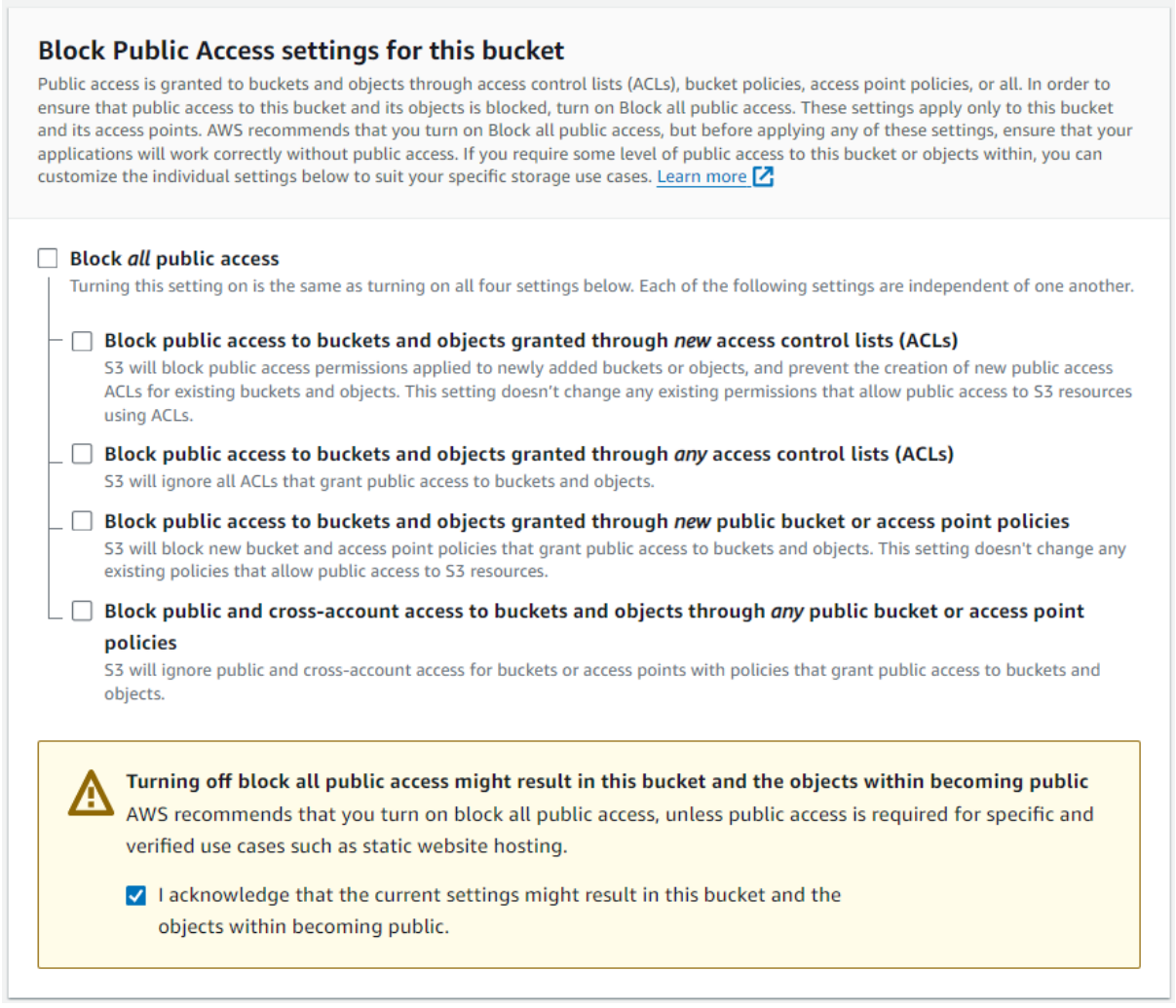
7. Once our source bucket is created now, we are going to create our production bucket.



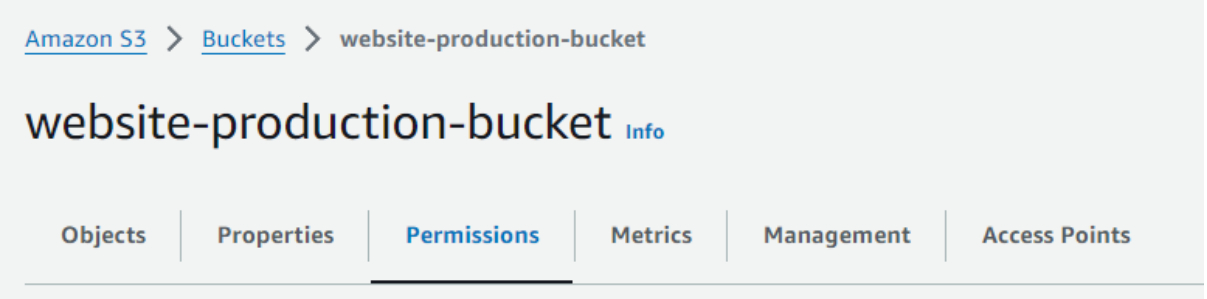
8. So, unlike source buckets, production buckets can be in any AWS region but for simplicity, we will keep our region same.

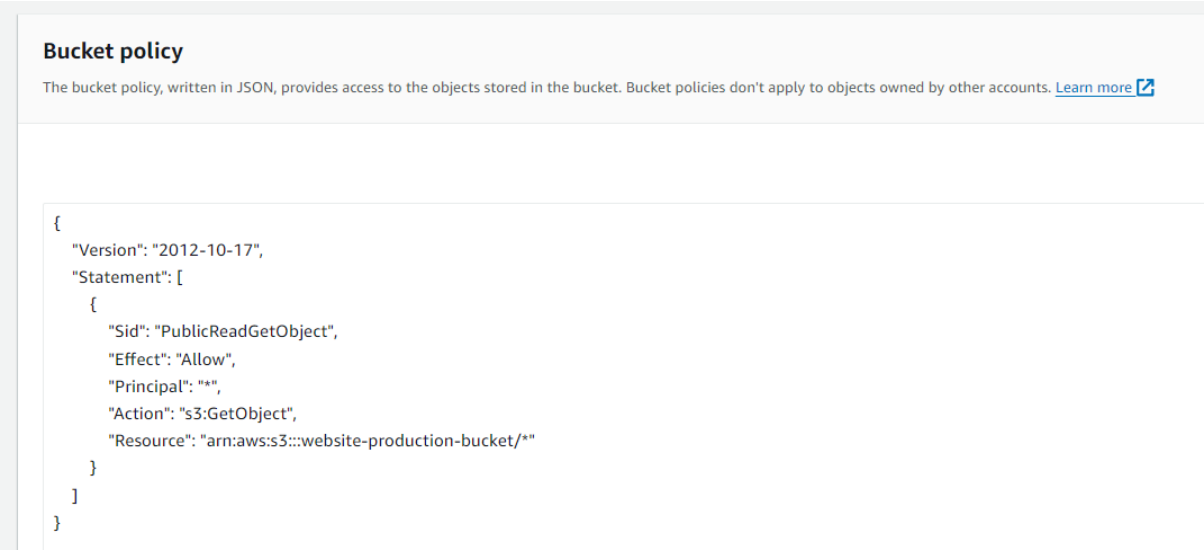


9. Our production bucket will be used to host a static website so, we need to disable block public access. Then scroll down to the bottom and create your bucket.



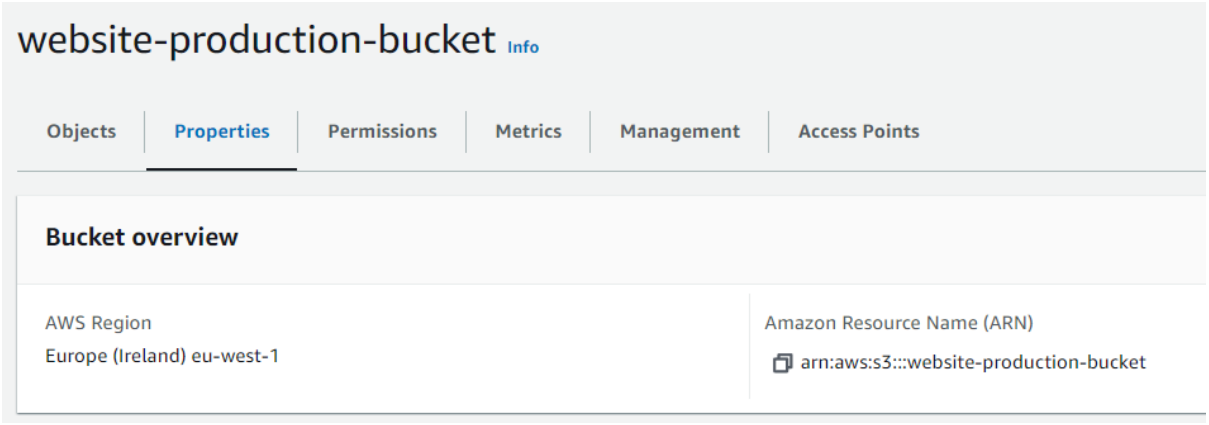
10. Now we need to go inside of our production bucket to update its bucket policy to allow its contents to be publicly readable from the Internet. So go to permission and scroll down to bucket policy click on edit then update the policy.

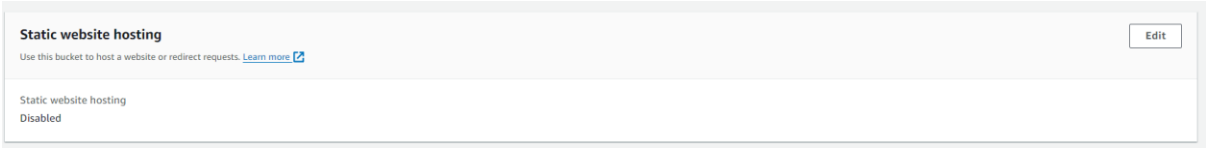


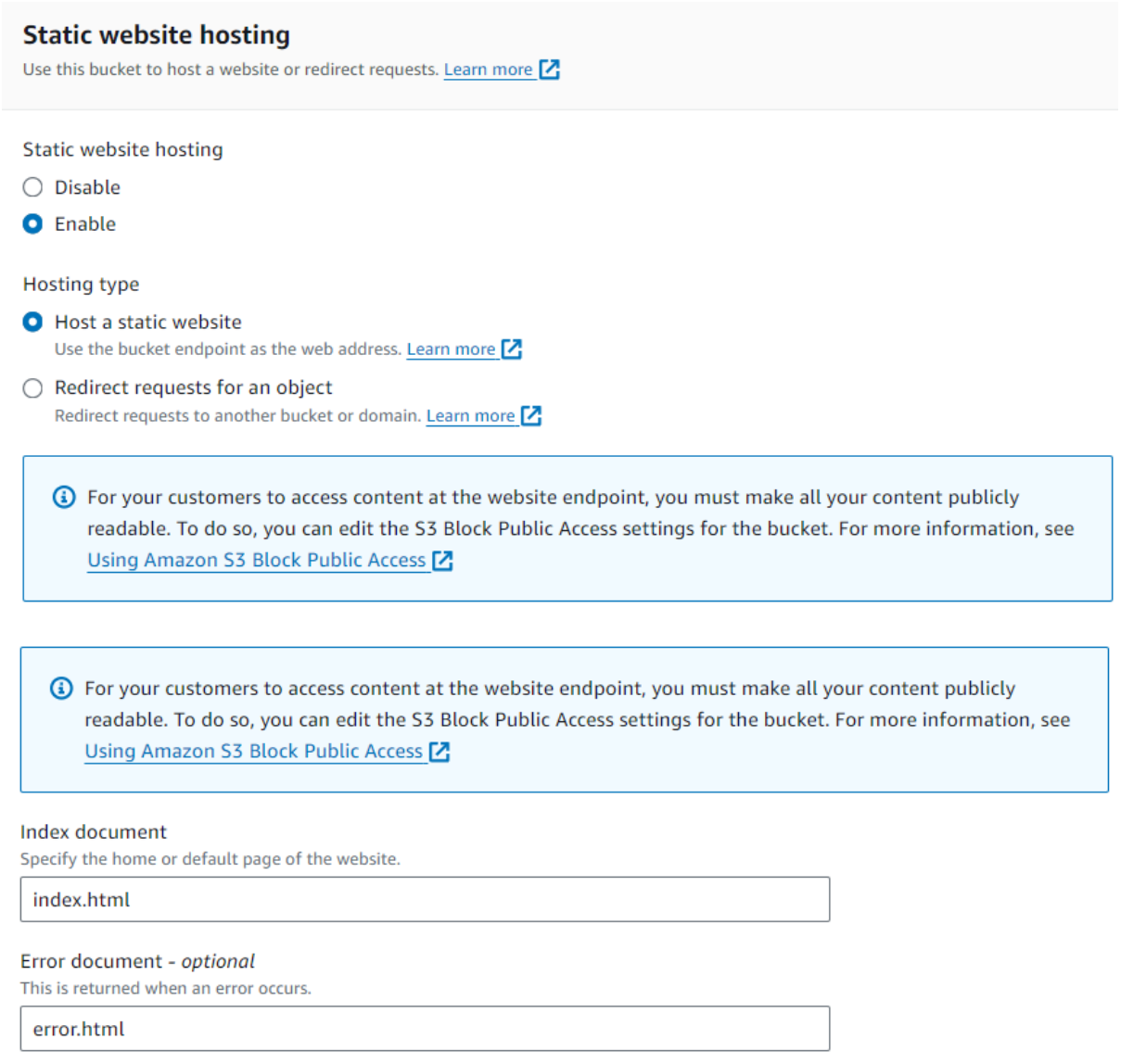



```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::YOUR-BUCKET-NAME/*"
      ]
    }
  ]
}
```

11. Now you need to go to its properties and then scroll down to the bottom there you will find static website hosting you need to enable this feature. You can follow the snapshots below to enable this feature.

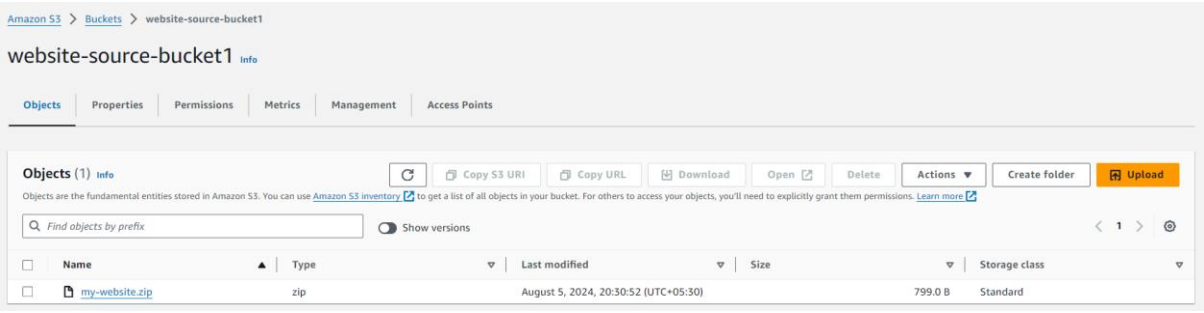




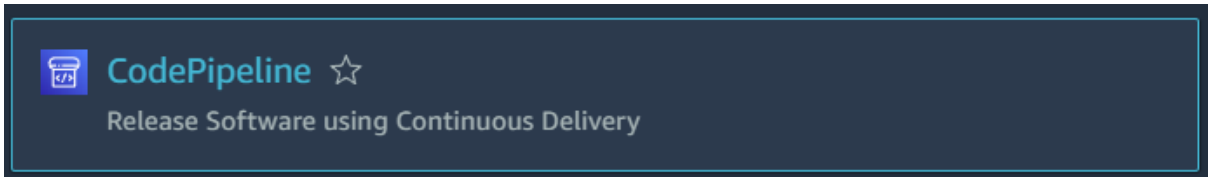


12. Now we are going to upload the zip file of our static website onto our S3 source bucket. Below you can see that we have uploaded the static website zip file.

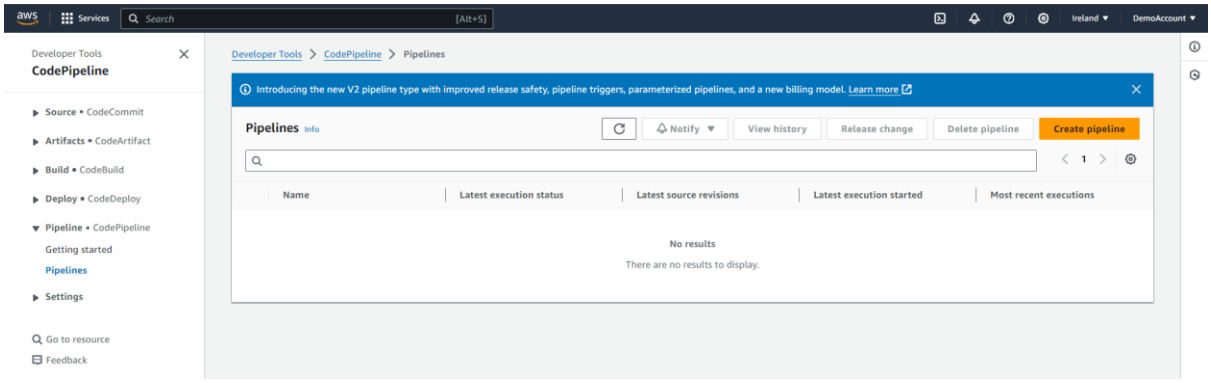
Note: You can get this zip file from the GitHub.

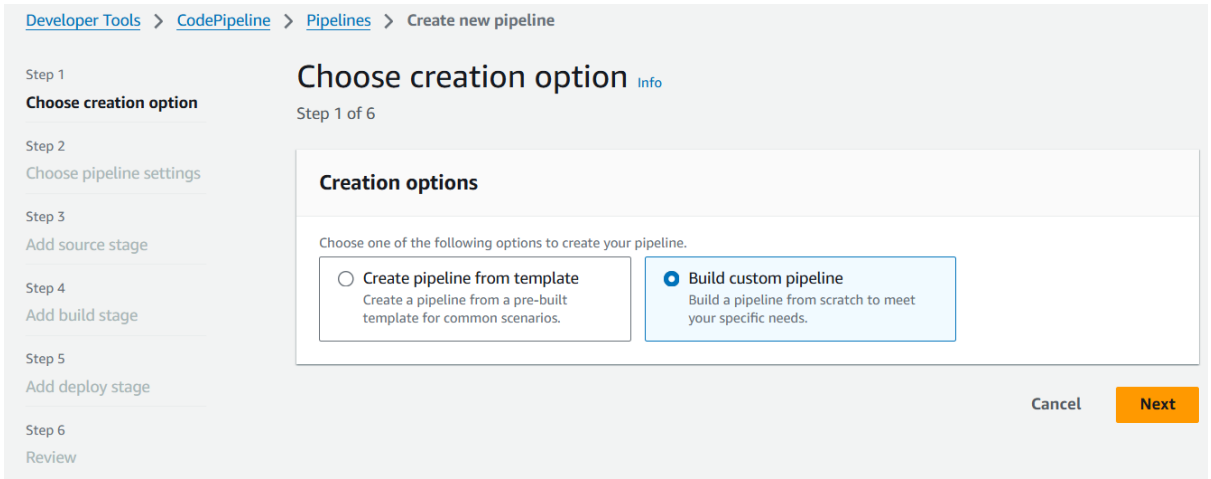


13. Now you need to search for Code Pipeline in your AWS Console and choose the service accordingly.

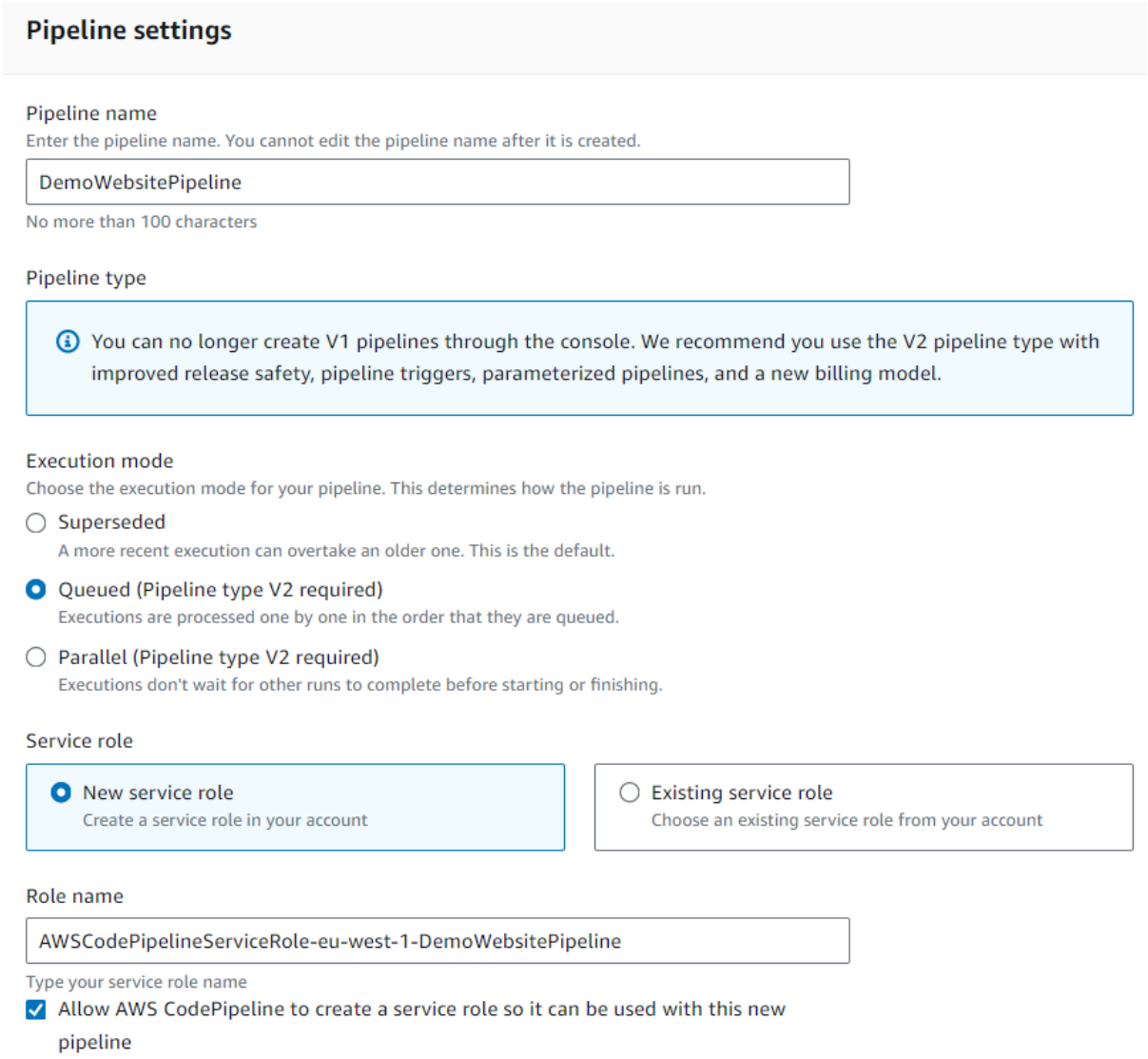


14. Here click on Create Pipeline then choose to build a custom Pipeline and click on Next.

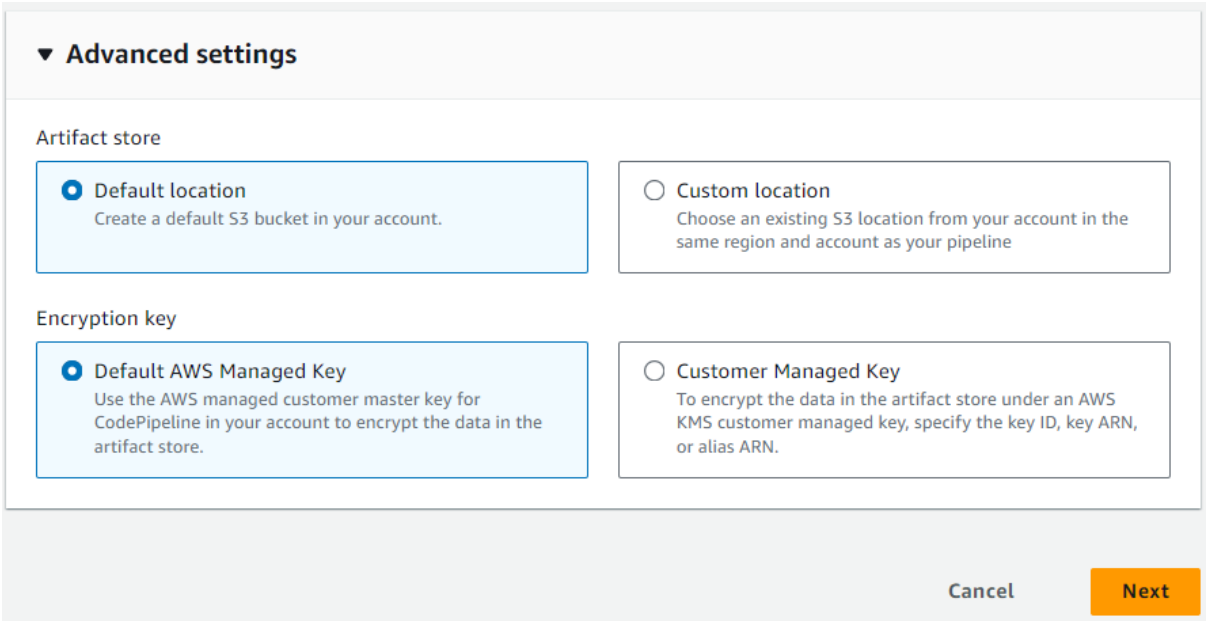




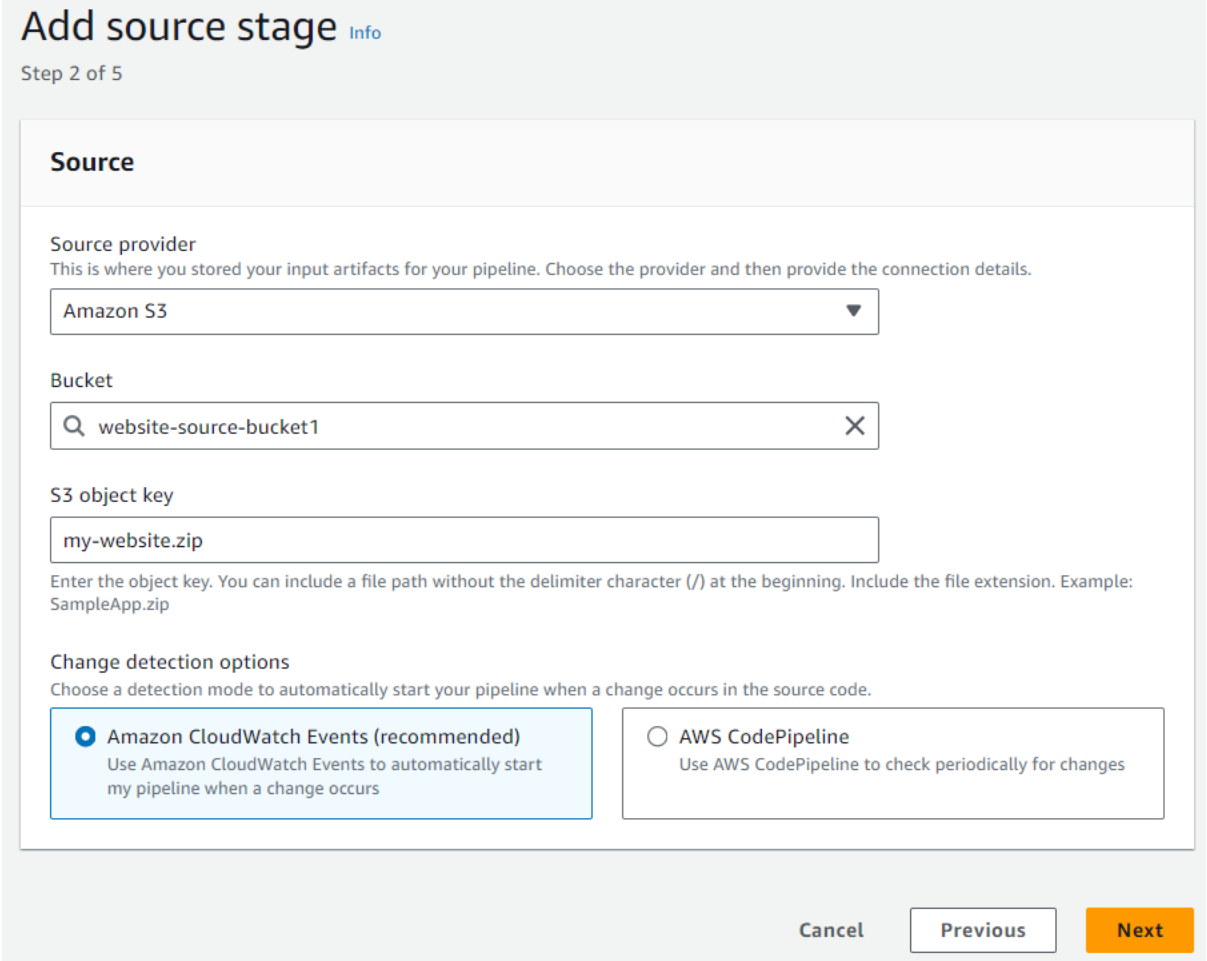
15. Now you just need to give your pipeline a unique name and keep other settings to default then scroll down to bottom.



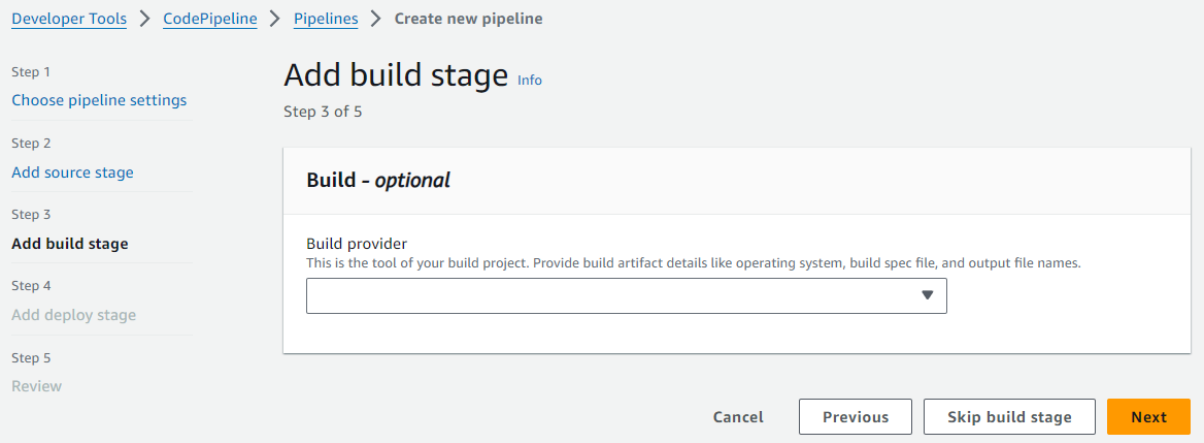
16. Then you need to expand the advanced settings options and here you will see that it will create an S3 bucket in our account to store artifacts. Also, we can give a custom location but let's leave it to default for now. Same for the Encryption key. Click on next.



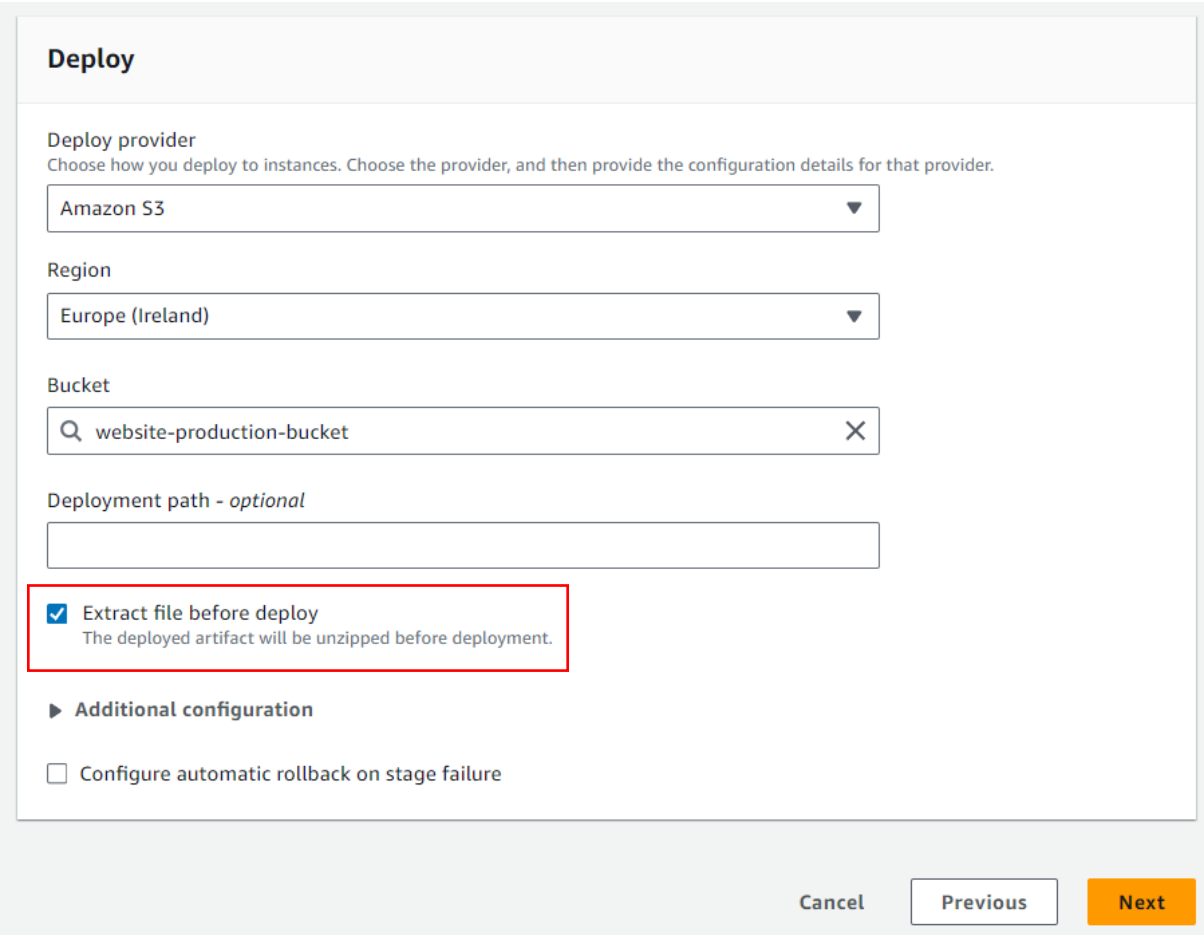
17. Now you need to add a source provider, so choose Amazon S3 and then choose your source bucket. Then you need to give the name of your zip file in the S3 object key. Keep the change detection option to Amazon Cloud watch events and click on next.



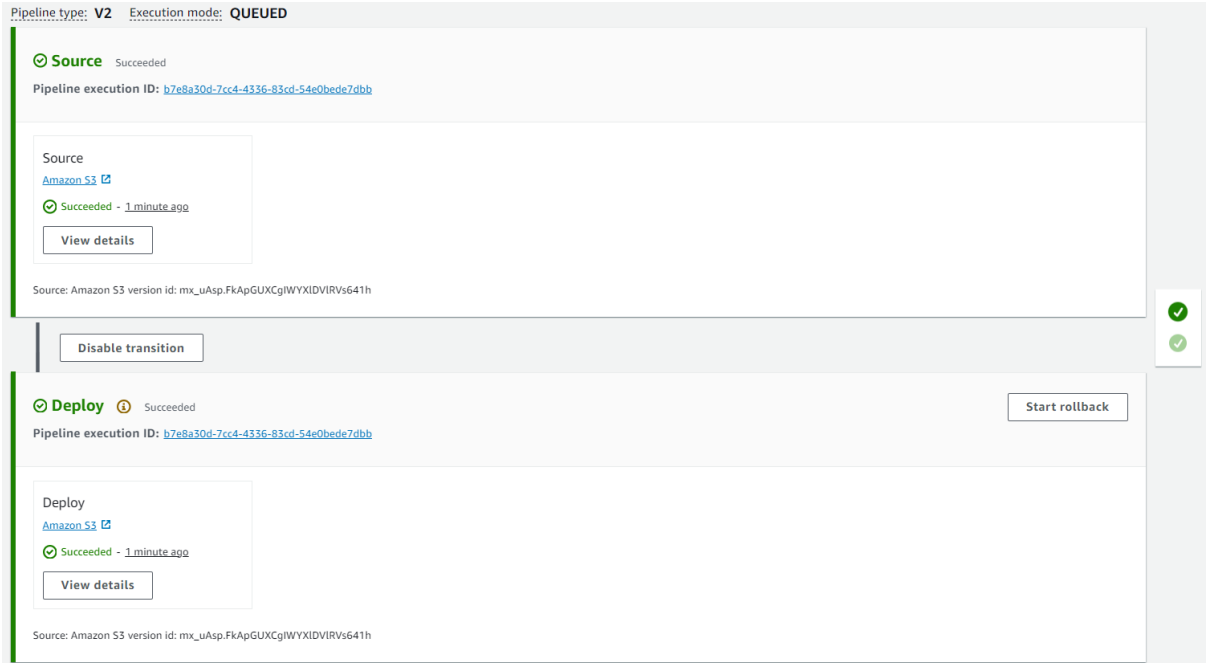
18. In the next option it is asking us to choose any build provider but we don't need any at this stage. So, choose the skip build stage and move to the next step.



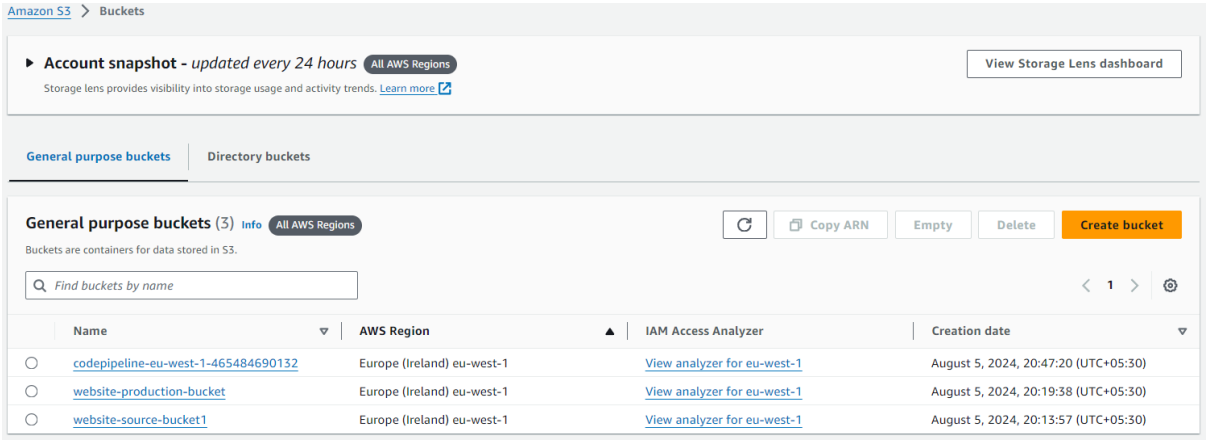
19. Now we need to choose a deploy provider, for that, you need to choose Amazon S3 and in the region choose the same region where you are creating the pipeline because we are not changing our region for now.
20. Then you need to choose the bucket and choose Extract file before deploying, click on next.
21. After moving to the review page create your pipeline.



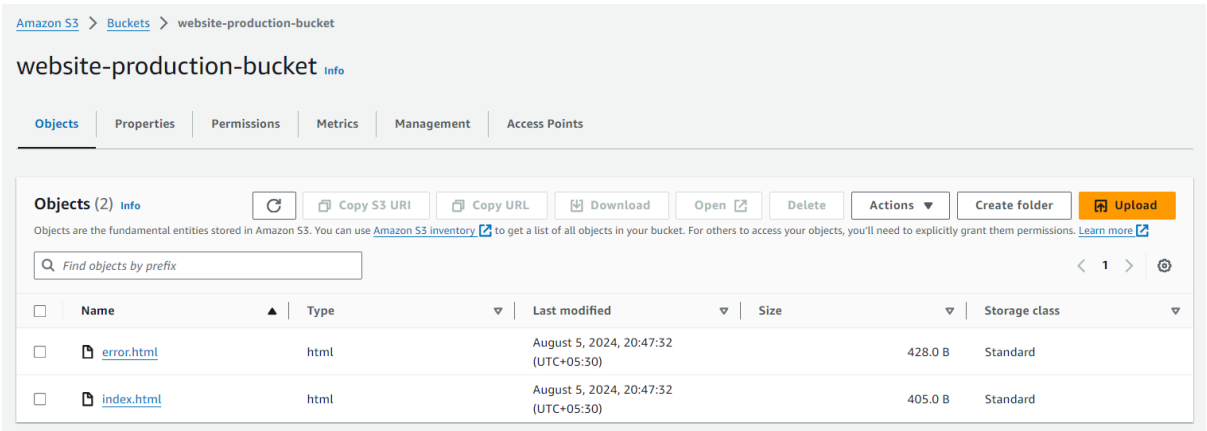
22. As you see, there are two large boxes named 'Source' and 'Deploy', which are connected with an arrow. These are the stages of our pipeline. When you create a pipeline, it is triggered automatically.



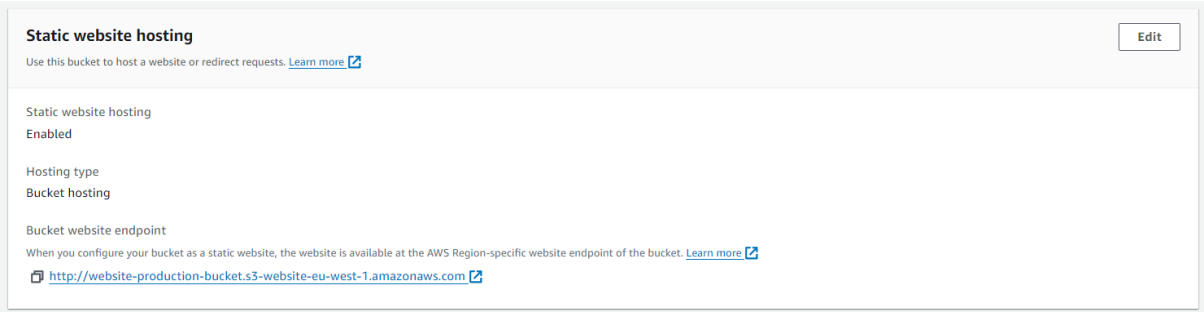
23. If you return to S3, here you will see that we have a new bucket which is created by the code pipeline.



24. Now go inside our production bucket, you will see that we have both of our HTML files.



25. Now go to properties and scroll down to the bottom here you will see the link to visit your website click on it.



26. Below you can see that our website is working fine and it was deployed successfully.