**😊 AWS EC2 Image Builder**

Amazon EC2 Image Builder is a service provided by Amazon Web Services (AWS) that simplifies the creation, maintenance, and distribution of customized Amazon Machine Images (AMIs) for use with Amazon EC2 instances. It allows users to automate the process of building secure and up-to-date images, which can include operating system patches, software configurations, and applications.

Here's a brief overview of how EC2 Image Builder works:

1. **Define Components:** You start by defining components such as operating system versions, software packages, security configurations, and scripts that you want to include in your images.
2. **Create Recipes:** Using these components, you create recipes that specify how to build your images. Recipes can include steps like installing software, configuring settings, and running scripts.
3. **Schedule Builds:** You can schedule builds to automatically create new images at regular intervals or in response to events like security updates.
4. **Monitor and Manage:** EC2 Image Builder provides monitoring and logging capabilities to help you track the progress of your image builds and troubleshoot any issues that arise.
5. **Distribute Images:** Once your images are built, you can distribute them to your EC2 instances or share them with other AWS accounts or regions.

**😄 Use Cases of EC2 Image Builder:**

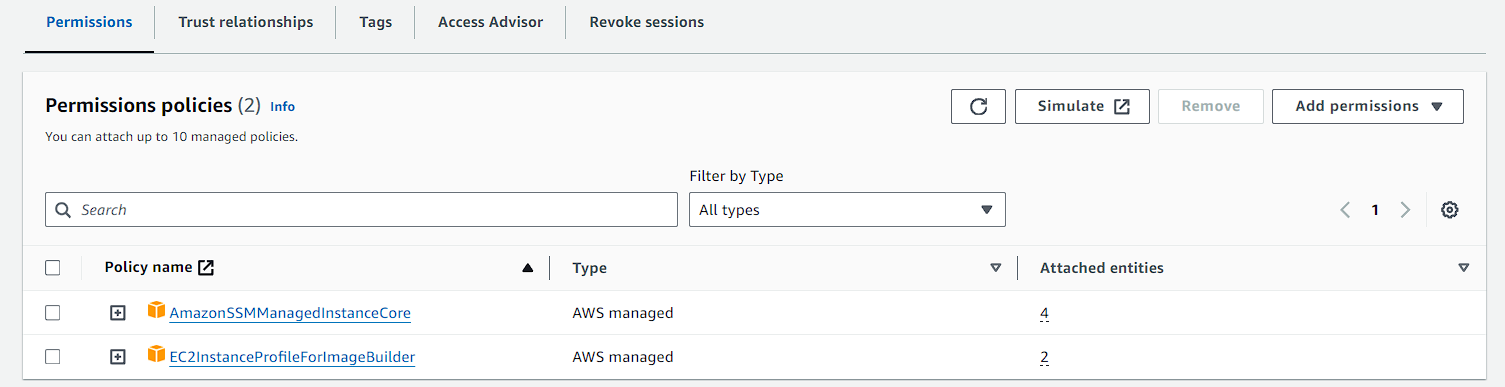
EC2 Image Builder offers several use cases that cater to different needs and scenarios within cloud infrastructure management. Here are some common ones:

1. **Standardized Application Deployments:** Organizations often need to deploy standardized configurations of operating systems and applications across multiple EC2 instances. EC2 Image Builder allows them to create custom AMIs with pre-configured software stacks, ensuring consistency and reducing deployment time.
2. **Security and Compliance:** Maintaining security and compliance standards is crucial for many industries. EC2 Image Builder enables organizations to regularly update and patch their AMIs with the latest security fixes and compliance configurations, reducing the risk of security vulnerabilities and ensuring regulatory compliance.
3. **Testing and Development Environments:** Development teams frequently require clean, consistent environments for testing and development purposes. EC2 Image Builder allows them to quickly create and deploy custom AMIs tailored to their specific requirements, speeding up the development cycle and improving productivity.
4. **Disaster Recovery:** In the event of a disaster or system failure, having up-to-date AMIs readily available can significantly accelerate the recovery process. EC2 Image Builder enables organizations to automate the creation and maintenance of disaster recovery AMIs, ensuring they are always prepared to restore critical systems quickly and efficiently.
5. **Multi-Region Deployments:** Deploying applications across multiple AWS regions often requires creating and managing AMIs in each region. EC2 Image Builder simplifies this process by allowing users to centrally manage AMI creation and distribution across multiple regions, ensuring consistency and reducing administrative overhead.
6. **Custom AMI Marketplace Offerings:** ISVs (Independent Software Vendors) and solution providers can leverage EC2 Image Builder to create custom AMIs tailored to their applications and services. They can then offer these AMIs in the AWS Marketplace, providing customers with pre-configured, ready-to-use solutions.
7. **Machine Learning Model Deployment:** For organizations deploying machine learning models on EC2 instances, EC2 Image Builder can automate the process of creating custom AMIs with the necessary dependencies and libraries for model inference. This streamlines the deployment of machine learning applications and ensures consistent environments for model execution.

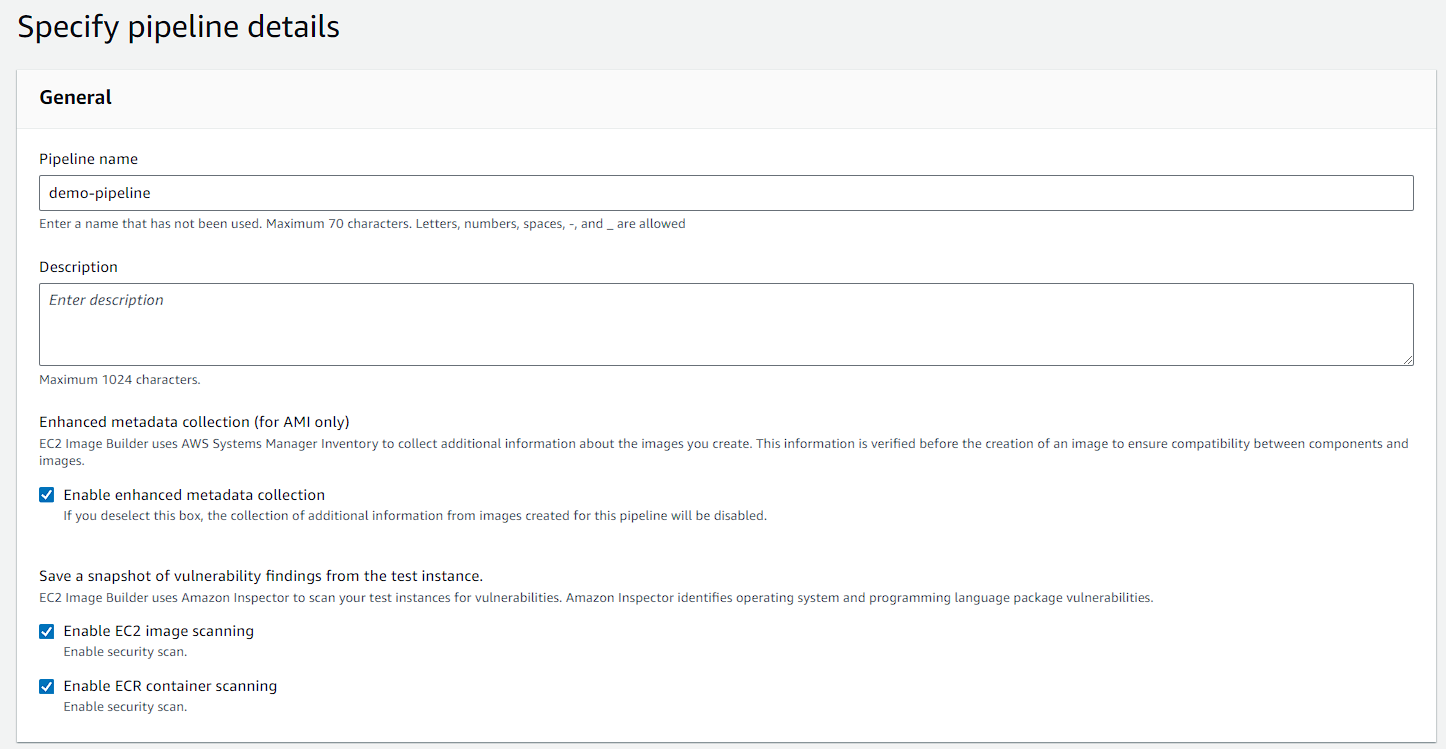
**In this guide, we're utilizing AWS EC2 Image Builder to automate the creation of custom Amazon Machine Images (AMIs) for EC2 instances. The end goal is to streamline the process of building, maintaining, and distributing standardized images that include specific configurations, software packages, and security settings. By defining components, creating recipes, and scheduling builds, we can ensure consistent deployments across multiple instances, improve security and compliance, accelerate development and testing cycles, facilitate disaster recovery, and simplify multi-region deployments. The step-by-step instructions provided walk through setting up a pipeline, running builds, testing images, and verifying the installation of components like the Amazon CloudWatch agent. Overall, the aim is to enhance efficiency, reliability, and consistency in managing cloud infrastructure within AWS environments.**

**😄 To begin with the Lab:**

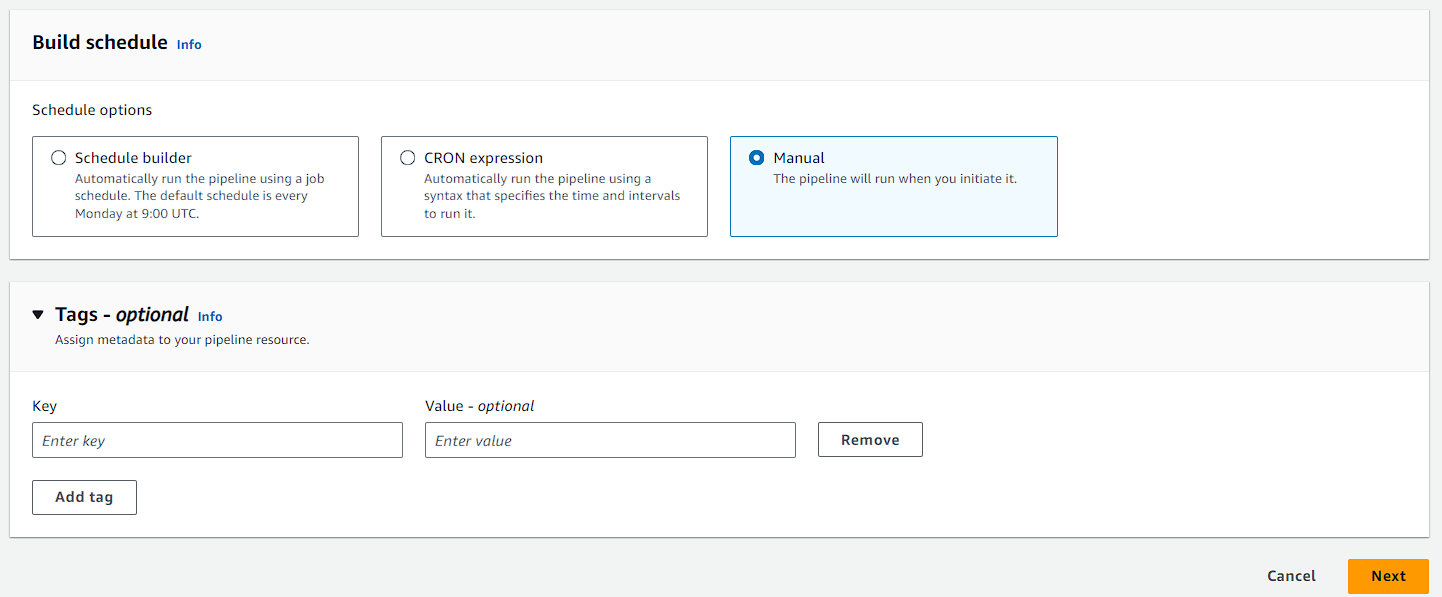
1. Log in to AWS Console then navigate to IAM, there you are going to create a role.
2. Now for this role you add these permissions mentioned below.



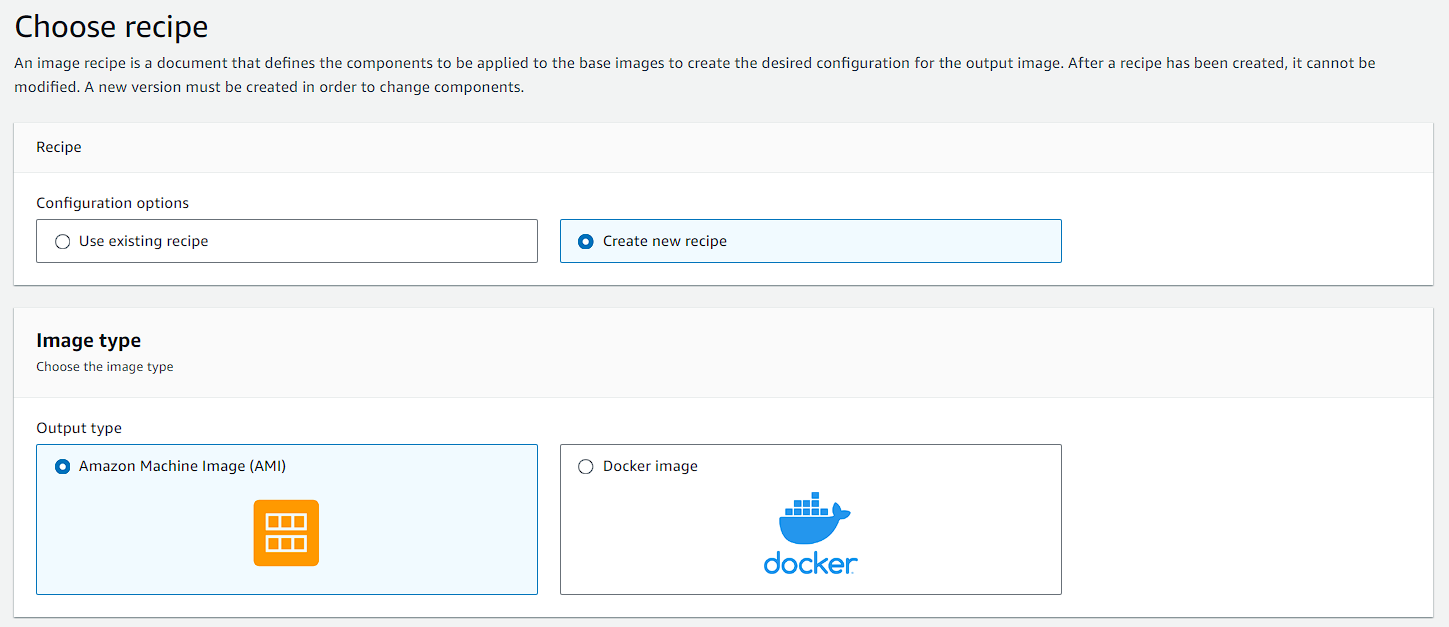
1. Then you are navigating to EC2 builder images. From there choose image pipelines and click on create.
2. On the first step you just have to give the name to your pipeline.



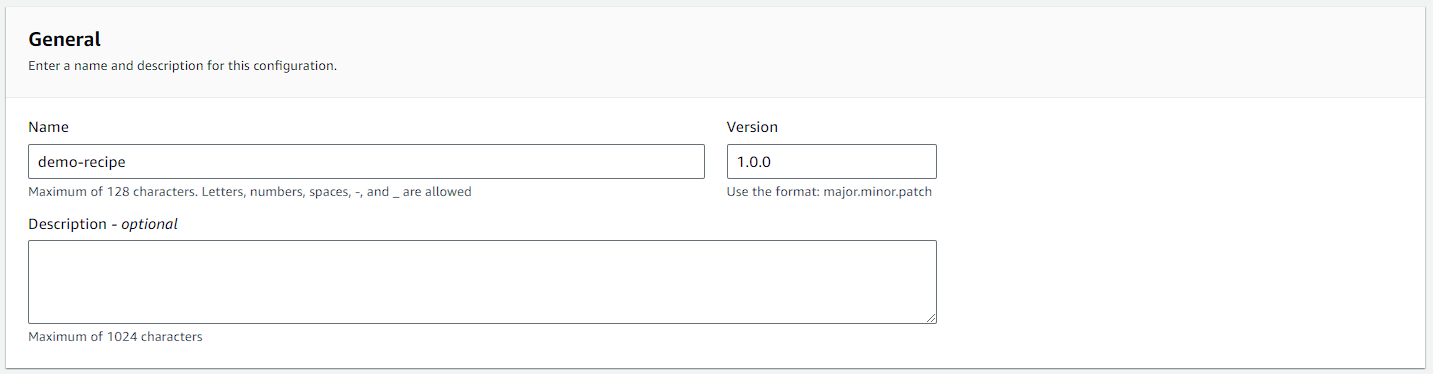
1. After that you have to choose Manual. This option will allow you to run the pipeline according to your desire. Now move to next step.



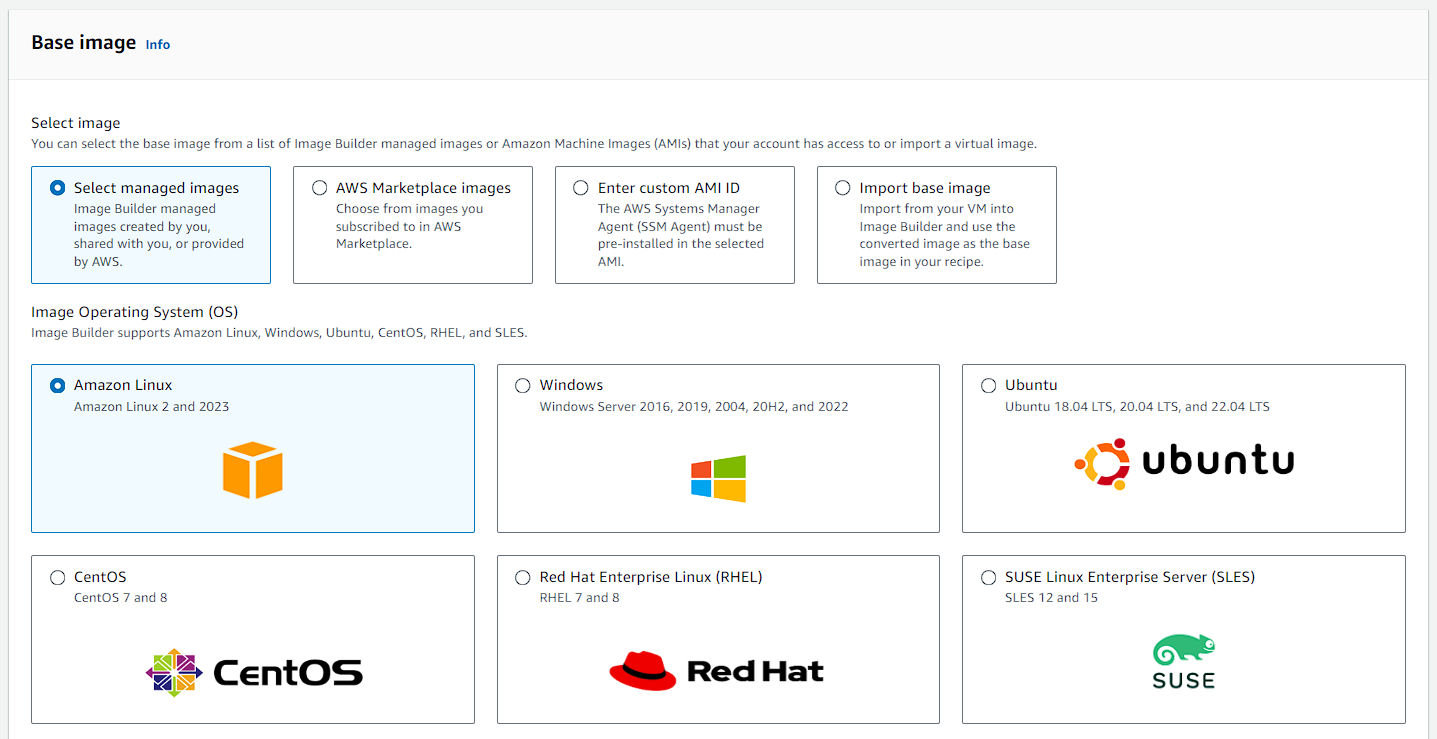
1. Here you have to choose a recipe and an instance type which is AMI.



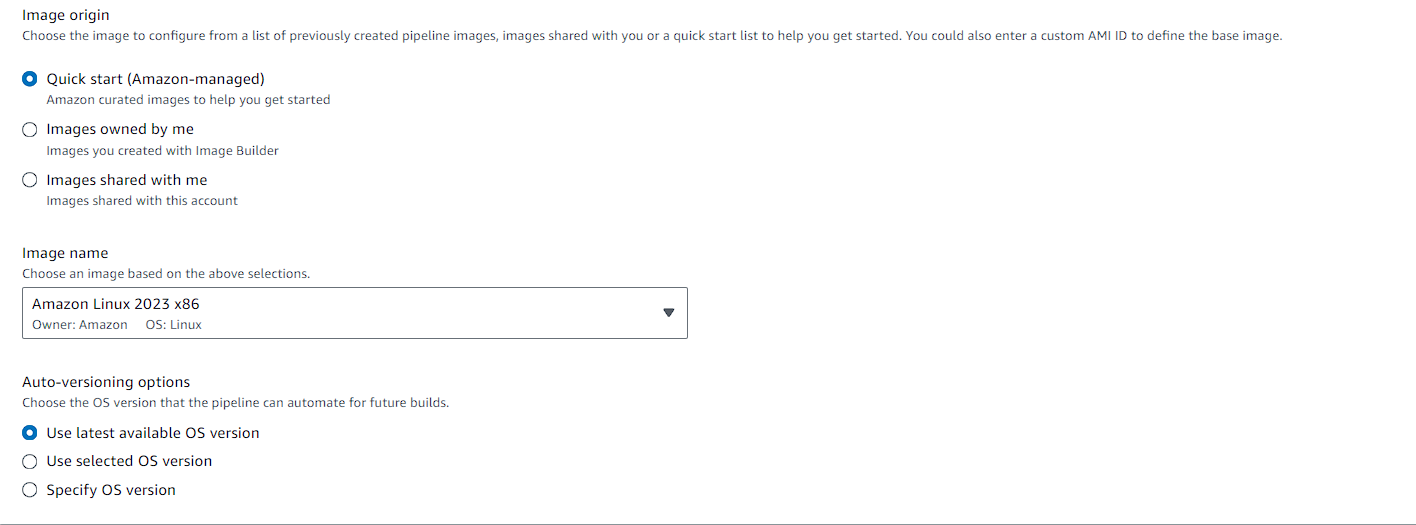
1. Then you have to give it a name and define it with a version.



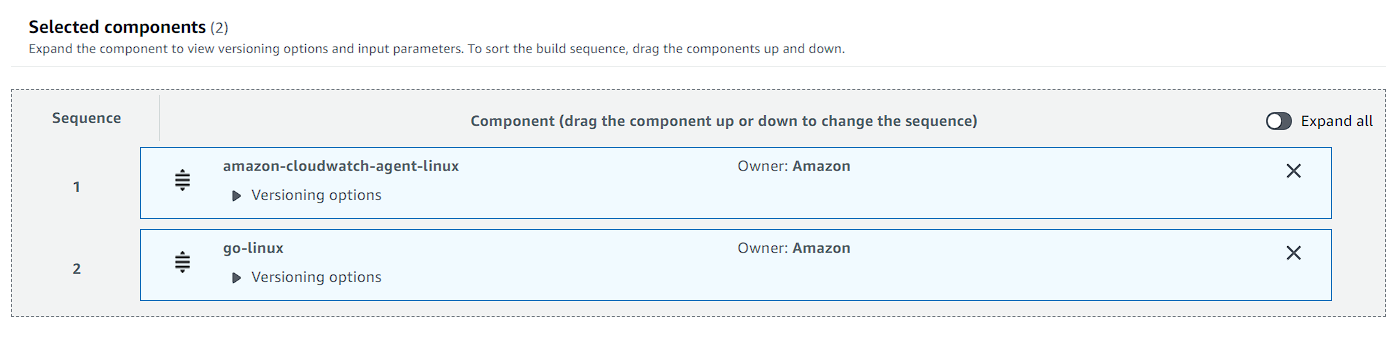
1. After that you have to select managed images and then click on Amazon Linux.



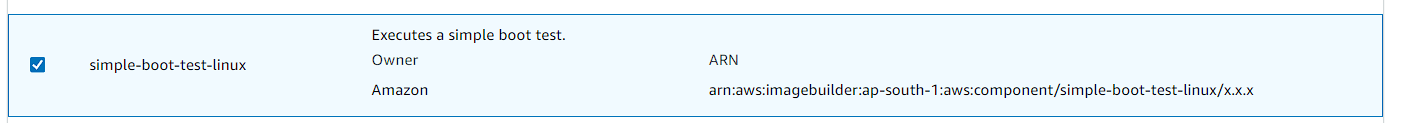
1. Then for the image origin choose quick start and for auto versioning use the latest.



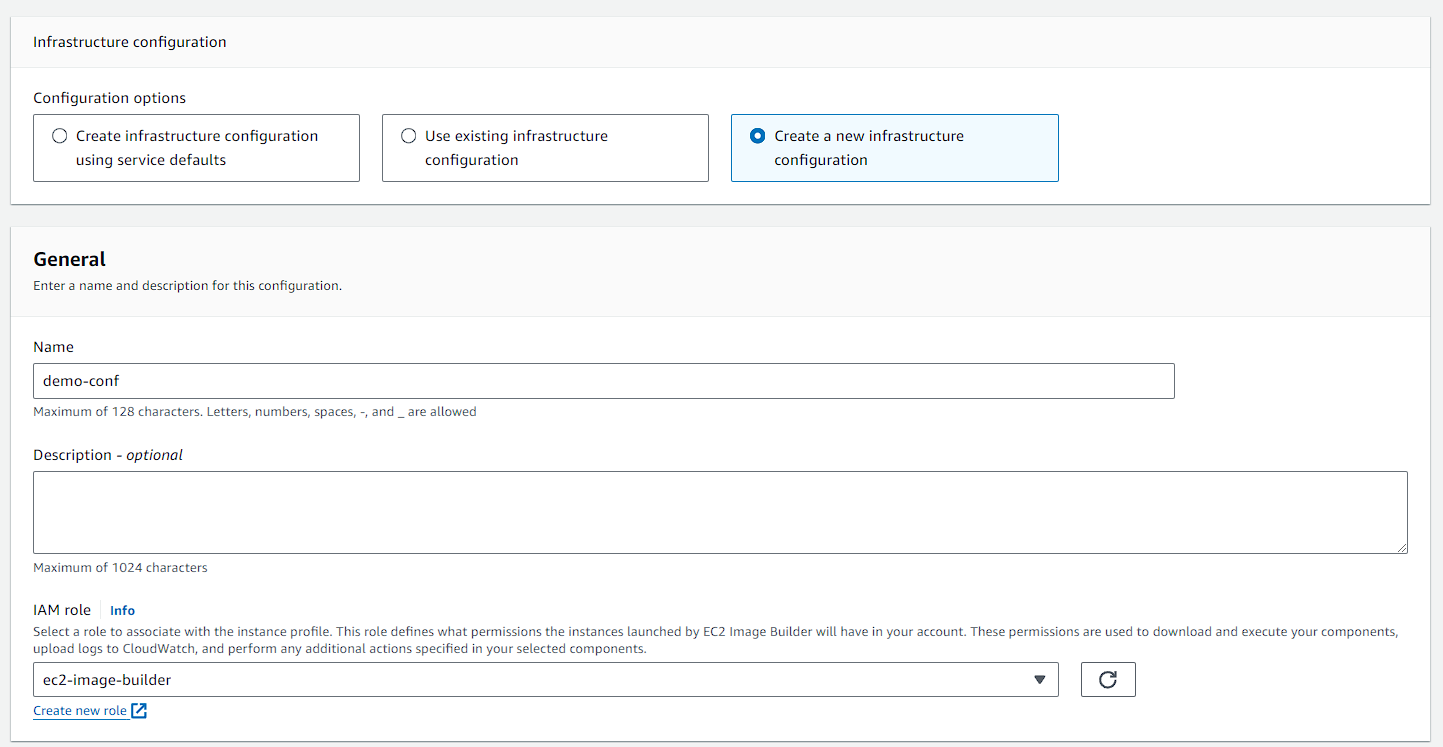
1. After that you have to come to the components and choose these 2 mentioned below.



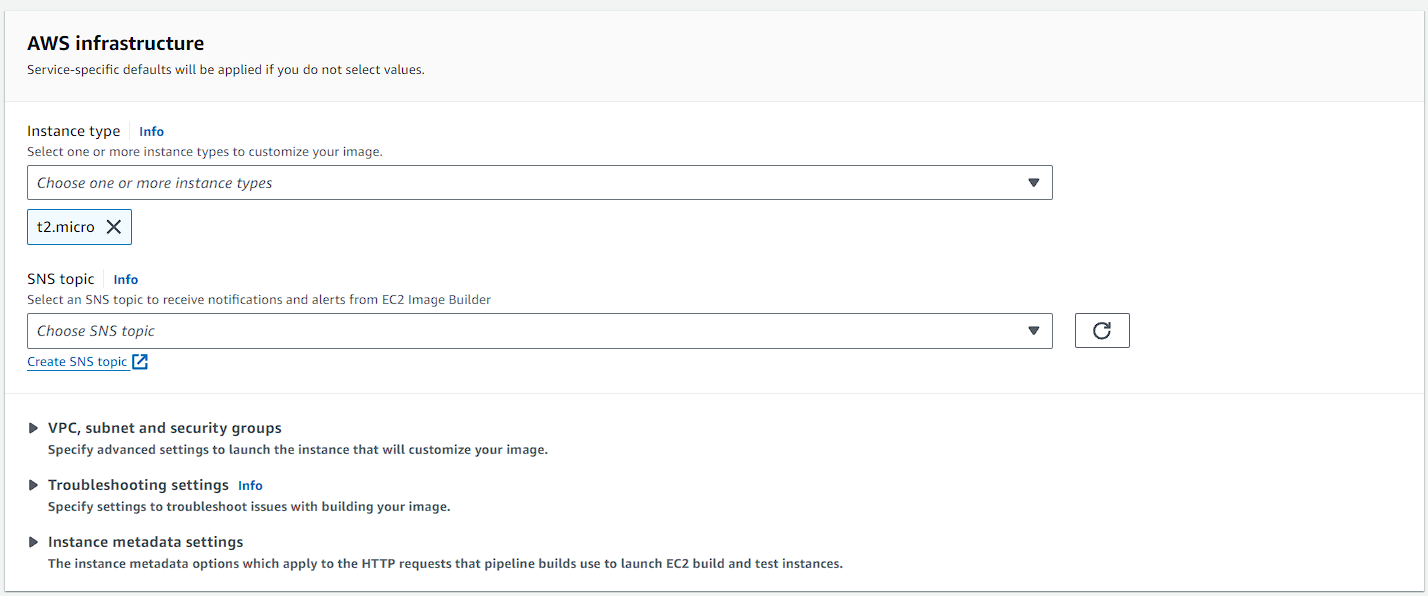
1. Now for the test components although it is optional you can choose a simple boot test. Then just move to next step.



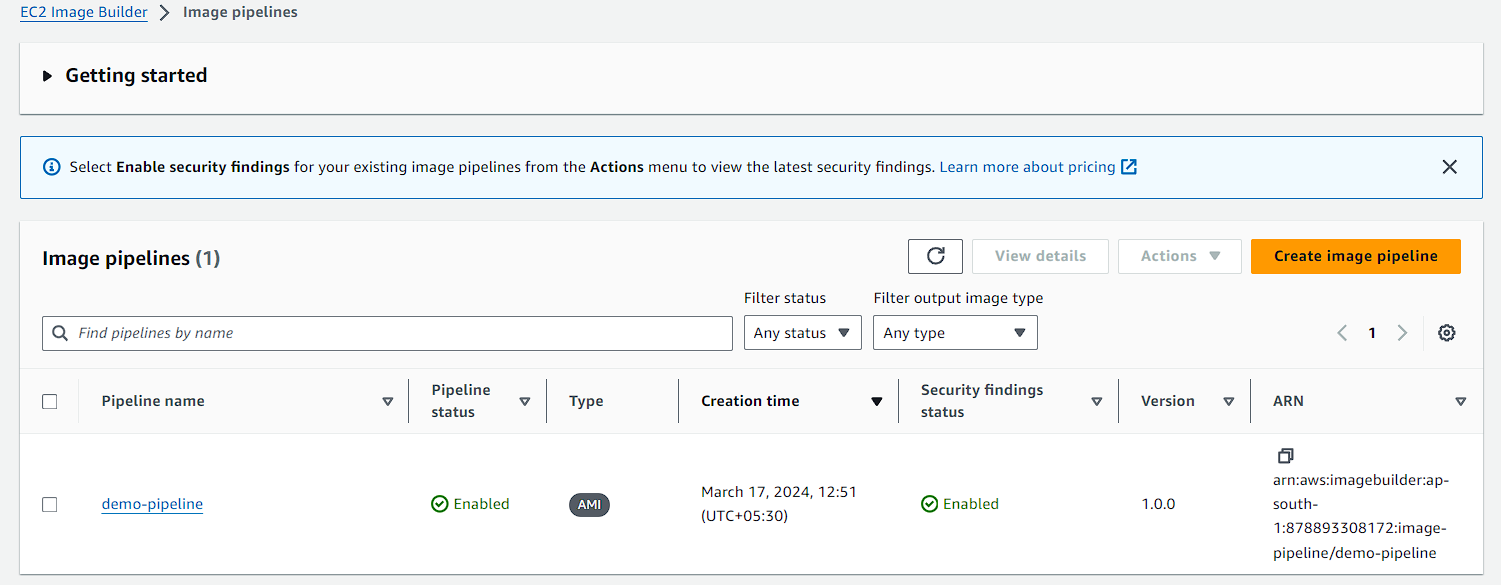
1. Keep step 3 as the default option and move to next.
2. First in the infrastructure configuration you have to choose to create a new one. Then give it a name.
3. After that you have to choose the IAM role which you created earlier.



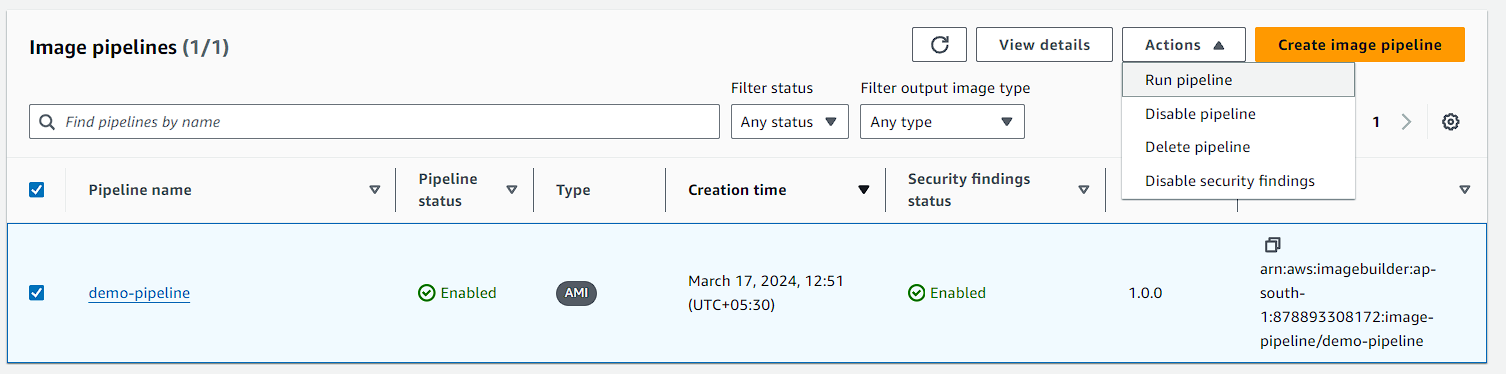
1. Now in the instance type keep it to t2.micro.



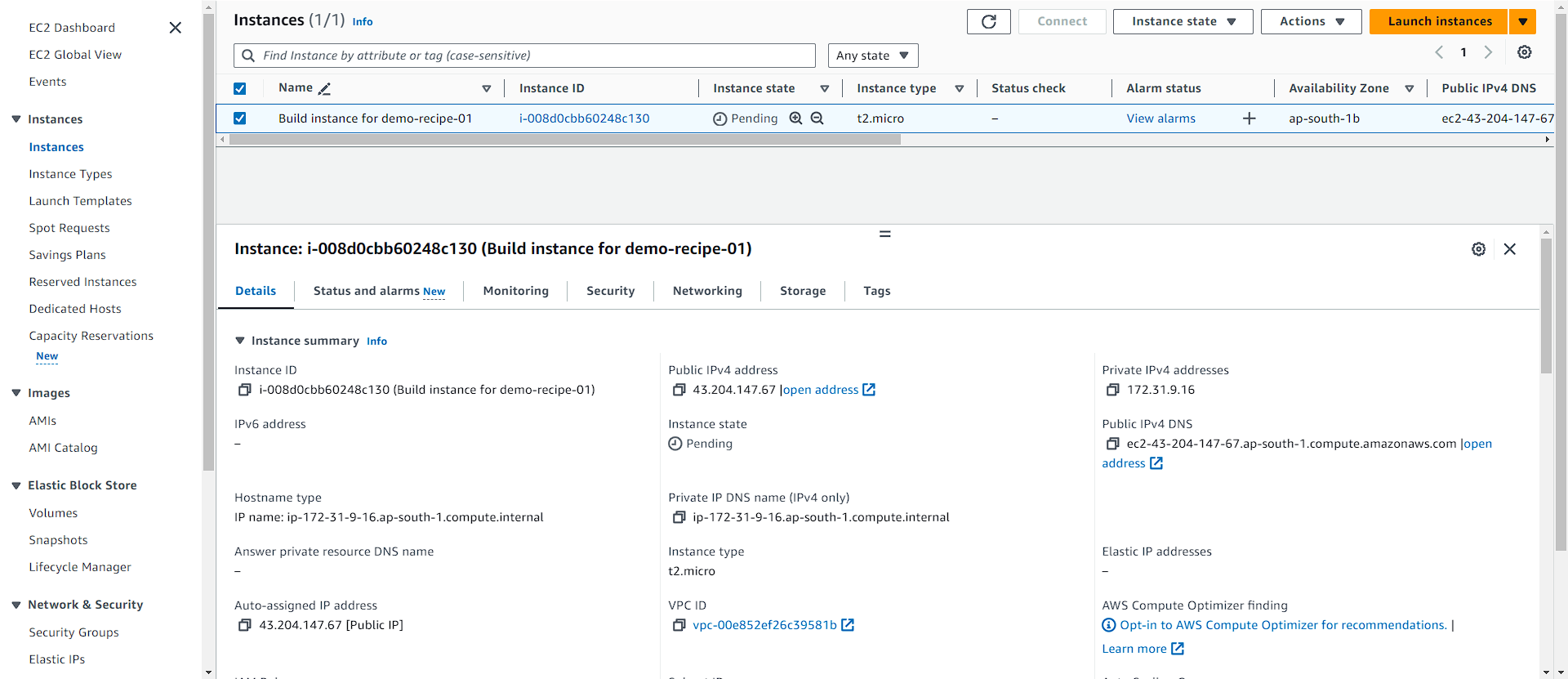
1. Now move to step 5, keep this step to default move to the review page, and create your pipeline.
2. After that you will be able to see your pipeline.



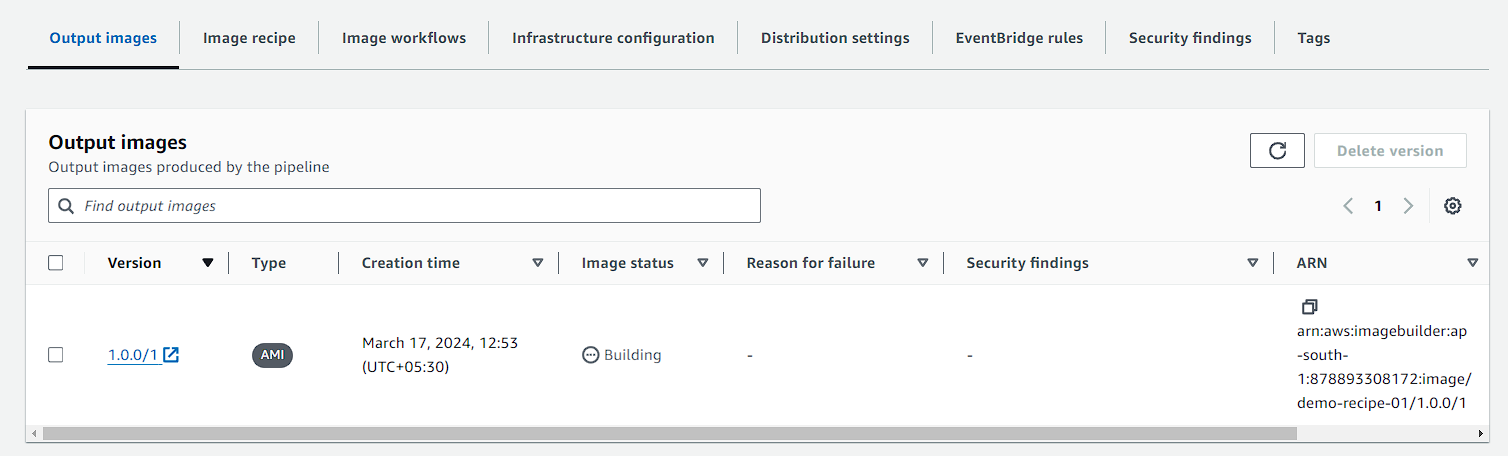
1. Now you have to select your pipeline and click on actions. Then click on run pipeline.



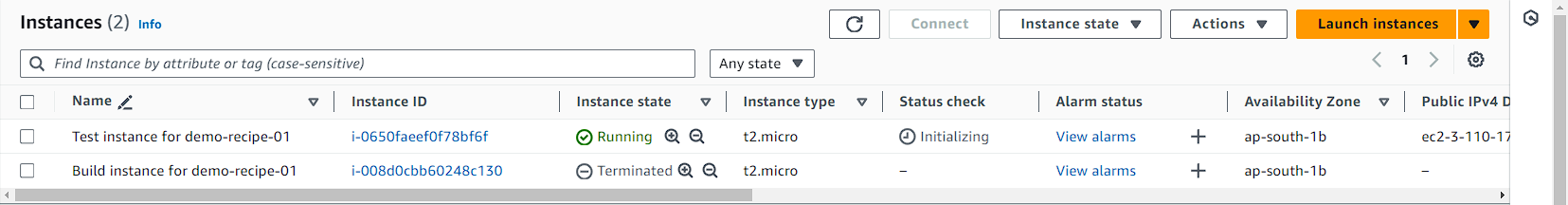
1. As soon as you will click on run pipeline, you have to navigate to EC2 and there you will see that an instance has been launched.



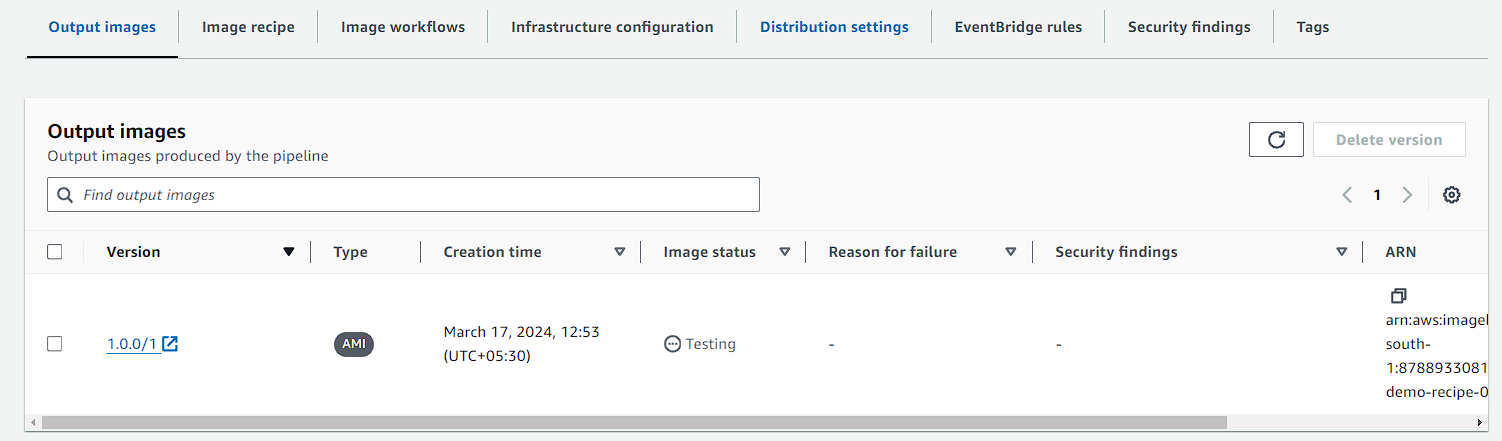
1. And if you open your pipeline, you can see that the version is building up.



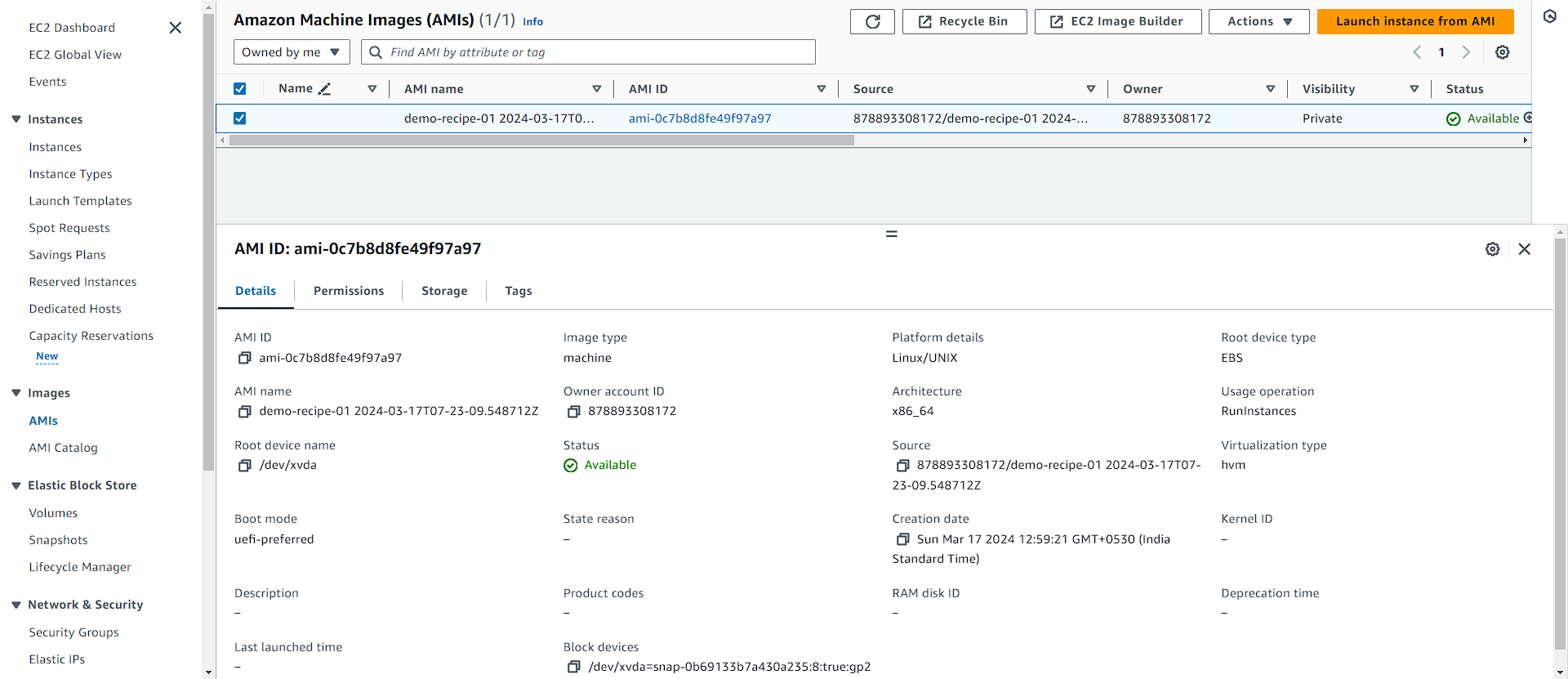
1. So, what the EC2 image builder will do first, it will create a new instance from the base AMI and it will configure all of the build components that we had added like Amazon CloudWatch agent and Go Linux. Once it adds it, it takes an AMI, it will test the AMI and our pipeline will get completed. So, it takes a little amount of time for the entire workflow to be completed.
2. After sometime if you’ll look again in EC2 you can see your builder instance is terminated and test instance is running.



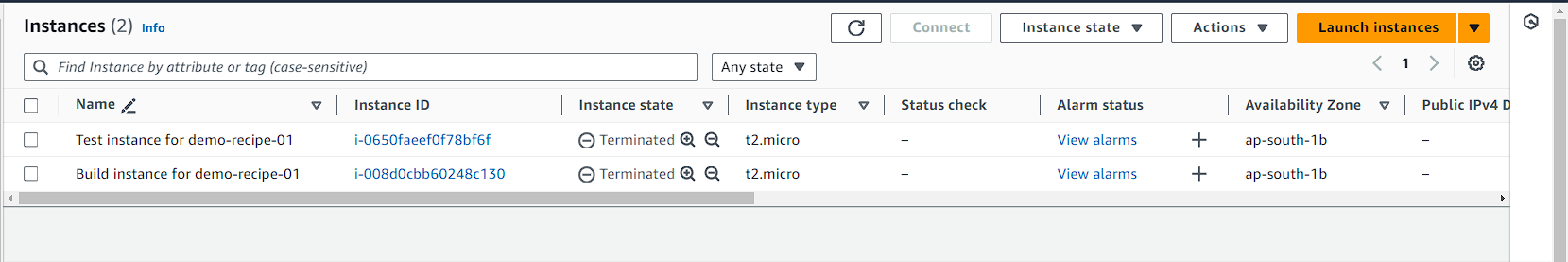
1. And if you’ll go to your image builder you can see that the image status is changed to testing from building.



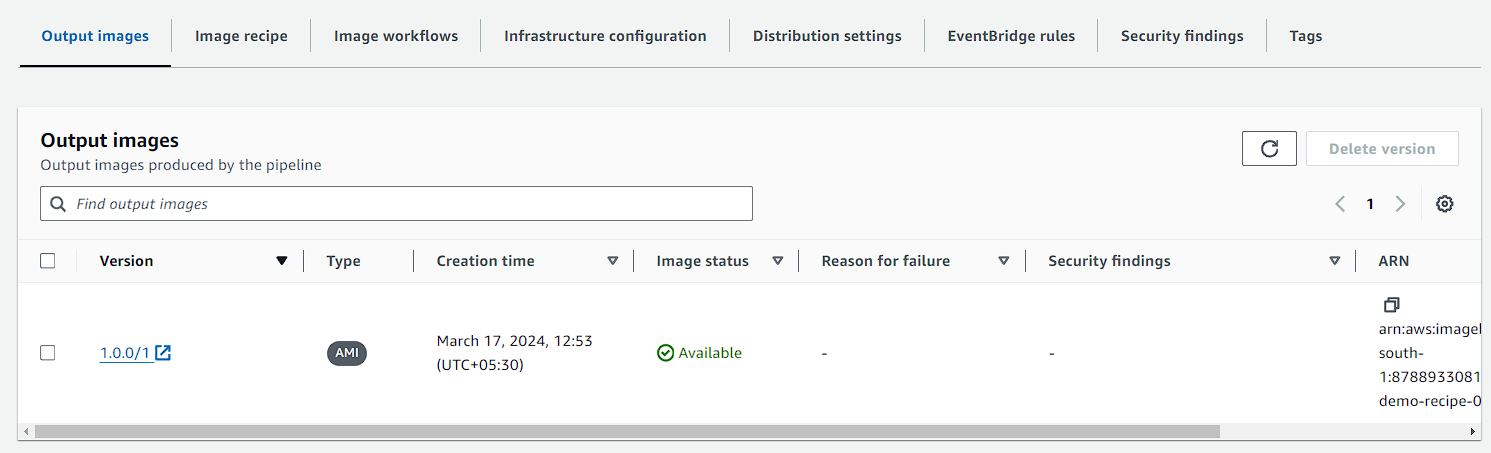
1. With that you can also see your AMI that has been created by image builder.



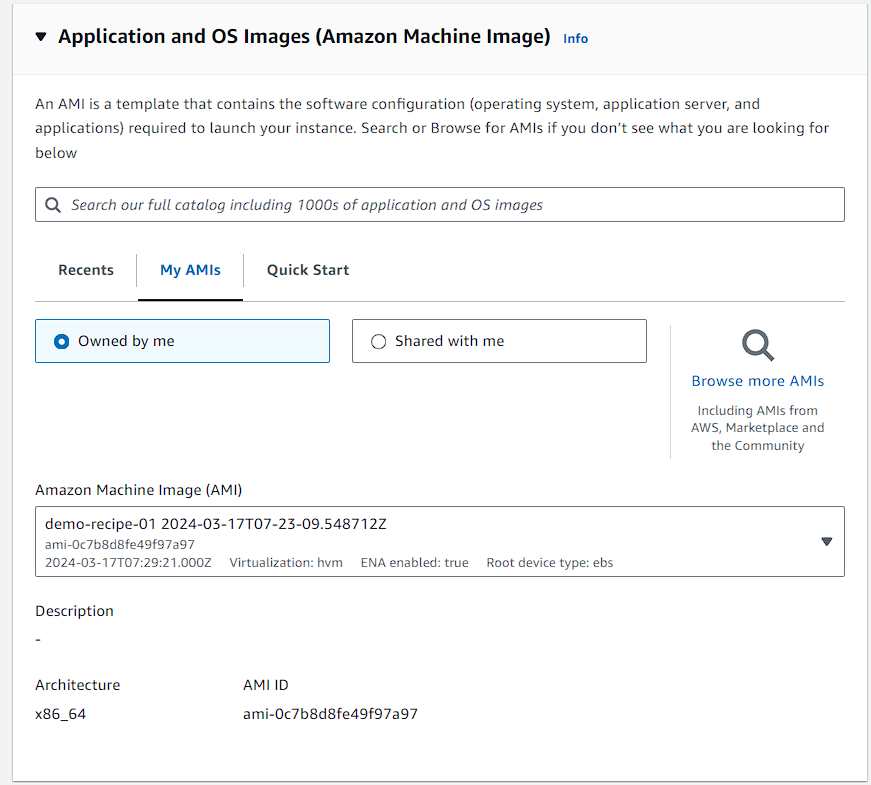
1. Now you can see that both of the instances are terminated which means that the testing and building are done.



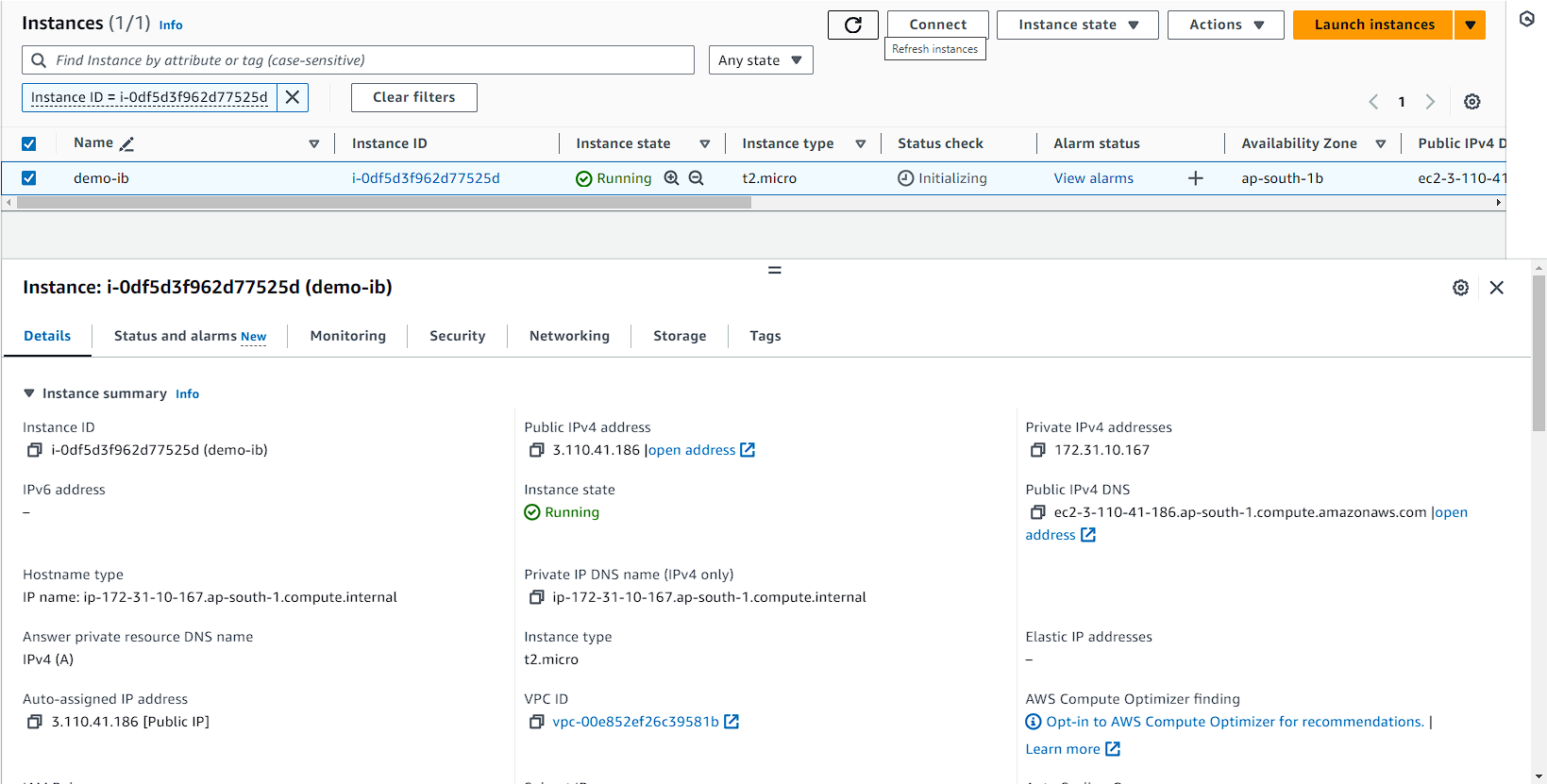
1. Now if you navigate to image builder in the output images you can see that the image status is available.



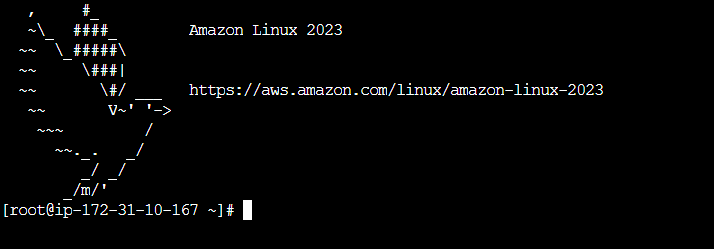
1. Now go back to EC2 and launch an instance. You can see that in the application and OS images sections, now you can choose your AMI which was created using image builder.
2. Now just launch an instance using this AMI.



1. Below you can see the instance has launched successfully.
2. Now try to connect via SSH.

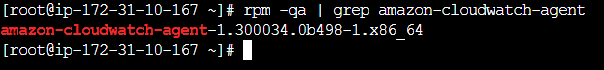


1. Here you can see your instance has connected successfully.



1. You can also check whether the CloudWatch agent is installed or not in your instance using the below command.

**rpm -qa | grep amazon-cloudwatch-agent**



1. There is one thing that is the image builder installs only the agent. You must take additional steps to configure and use the Amazon CloudWatch agent.