# Deploying Applications on Elastic BeanStalk Using Deployment Policies

Created by[Zeal Vora](https://deloittedevelopment.udemy.com/user/cybercorp/)

**AWS Workspace**

**60-120 minutes**

In this lab, you will be a consultant at an E-Commerce based startup that wants to create a simple solution which can quickly deploy the applications of their developers without them having to manually create and manage underlying resources like EC2, ELB, Auto-Scaling groups,, Web Server configuration, Deployment Configurations, and others. The architecture group thinks that using the Elastic Beanstalk service will allow developers to easily orchestrate the backend infrastructure while the developer can just focus on uploading their application. Your boss wants you to build a proof-of-concept.

The Engineering team at a company has raised an issue. For their application to be deployed, the developers have to manually create, configure, and troubleshoot various AWS resources like EC2, Elastic Load Balancers, Auto-Scaling groups, Web-Server configuration, and an application deployment pipeline. This is a time-consuming task for the developers, and they would just like to focus on their application instead of managing infrastructure. I want you to create an environment where a developer can simply upload their application, and the rest of everything is taken care of by the environment. It is also necessary to maintain the full capacity of servers serving production traffic even while new deployment happens. Any failed deployment should not hamper any existing servers serving traffic.

The environment should be based on Elastic Beanstalk. The platform should be based on Docker. All that the developers must be required to do is to upload their application zip file, and the rest process needs to be taken care of by the service automatically. You need to demonstrate two use-cases where:

1. When a default Elastic Beanstalk environment is created with custom web-application, how the production environment is impacted when a failed deployment occurs.
2. When a specific different deployment policy is used based on the requirement of this lab, how it helps to mitigate the issue of a failed deployment.

Developers have provided you with two application files to simulate the scenario. The base-eb-app.zip represents an application based on the Docker platform that will get deployed successfully and an intentionally error-prone application filed named eb-failed-app-code.zip code that will lead to a failed deployment when uploaded.

**How you'll work**

Your project has been broken into a set of tasks. To complete these tasks, use the provided workspace. You can launch your workspace by clicking below or using the button in the top right of the screen.

Tasks

1-Creata a basic Elastic Beanstalk application and environment

2-Deploy an Error-ProneApplication

3-Configure the Deployment Policy

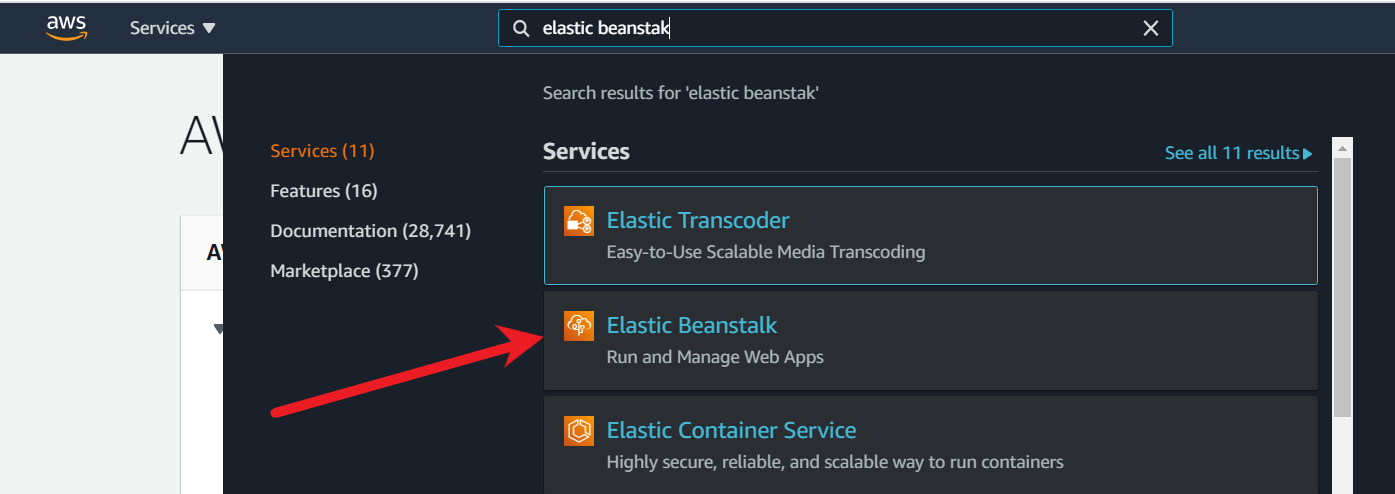
4-test the New Deployment Policy

5-Terminate the environment

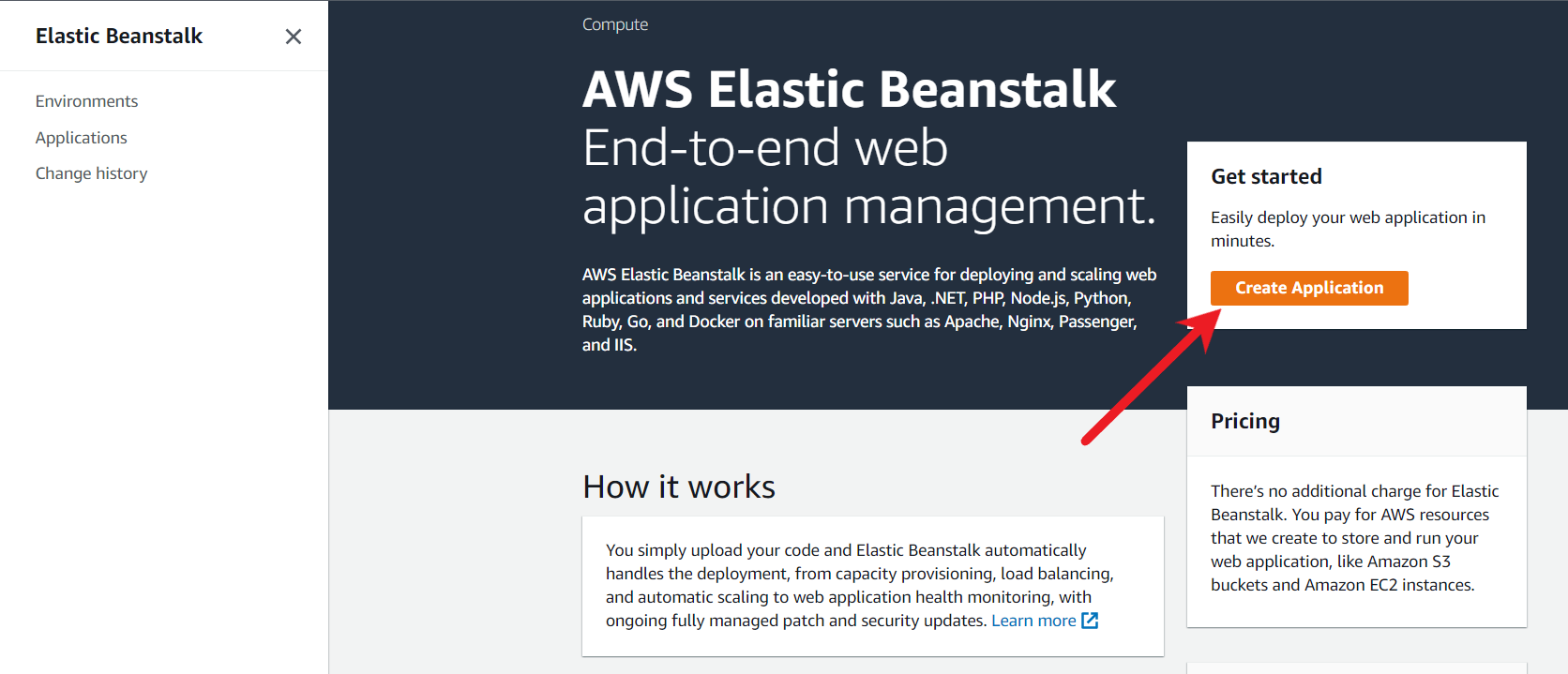
Creata a basic Elastic Beanstalk application and environment

You need to create a new application in Elastic Beanstalk and upload the sample code provided. The code is based on the Docker platform and the appropriate environment needs to be configured accordingly. You will need to upload the code and make sure the application is running.

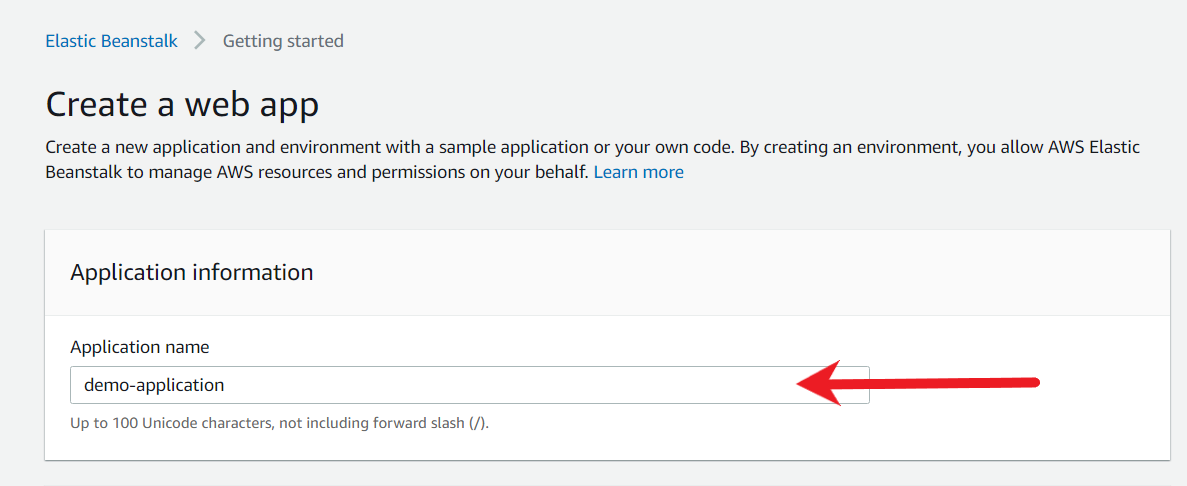
1. From the AWS console, search for the Elastic Beanstalk service and click on the available option.



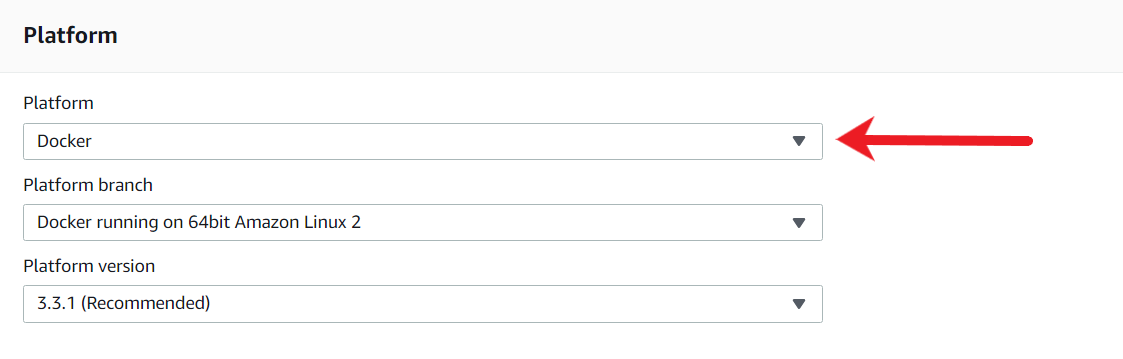
1. In the main console page, select **Create Application**



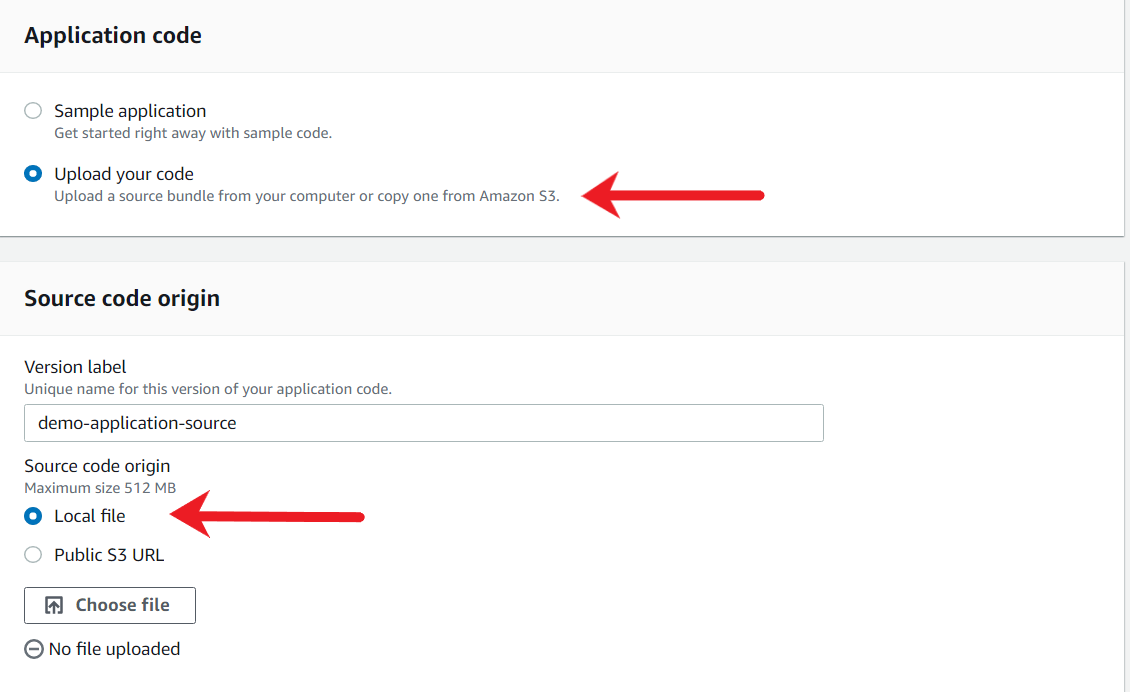
1. For **Application name**, give an appropriate name. For this demo, I will give the name as as demo-application



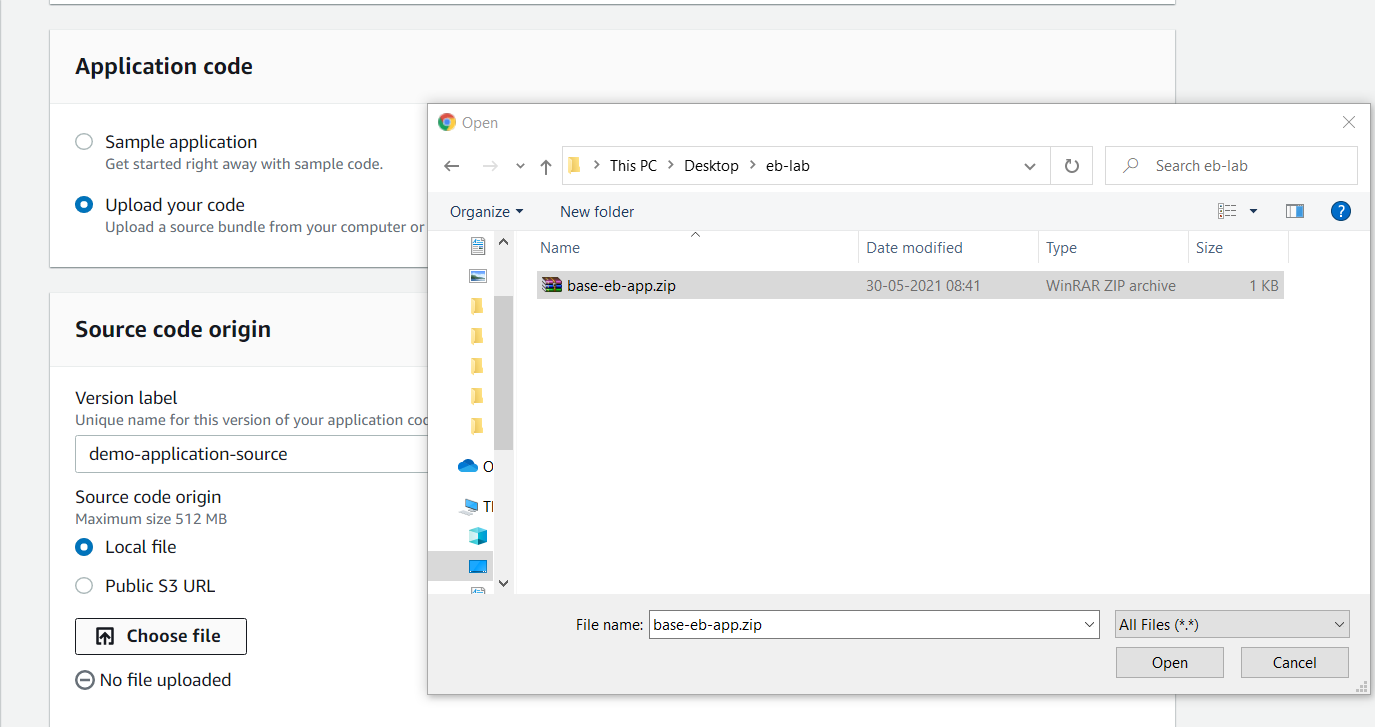
1. For **Platform**, select Docker. When you select a platform, the recommended version within the branch is selected by default.



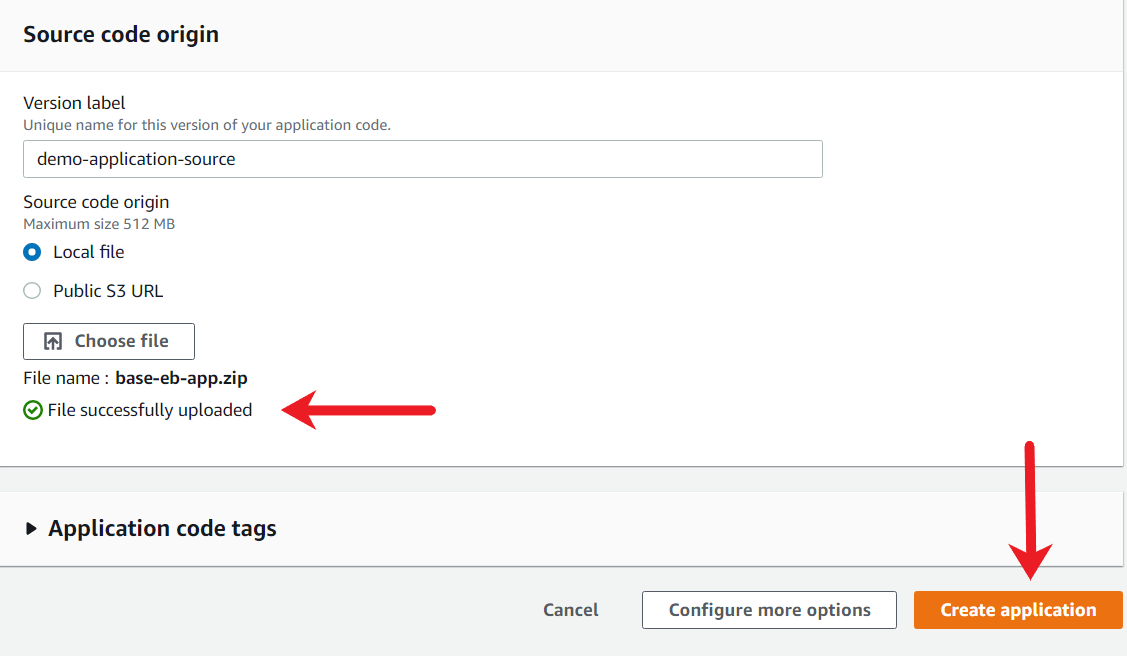
1. For **Application Code**, choose "Upload your code" option and the source code origin should be "Local File".



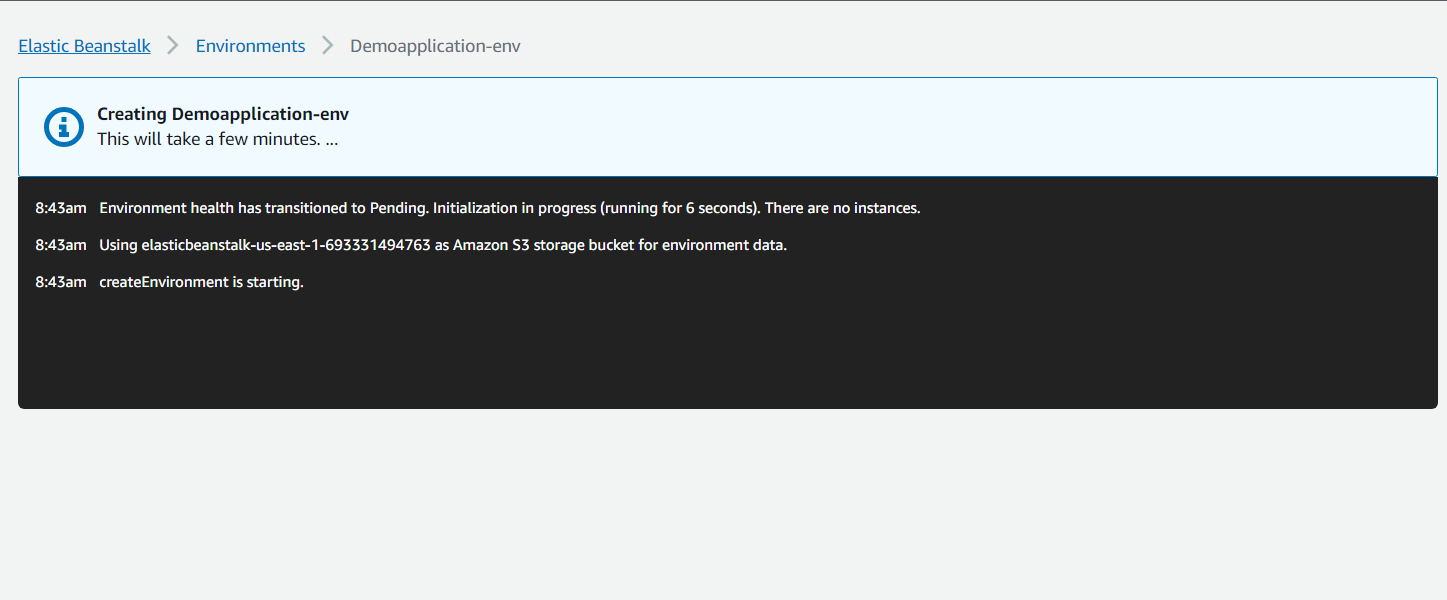
1. Click on the "**Choose File**" button and upload the base-eb-app.zip file that is provided to you in the lab assets.



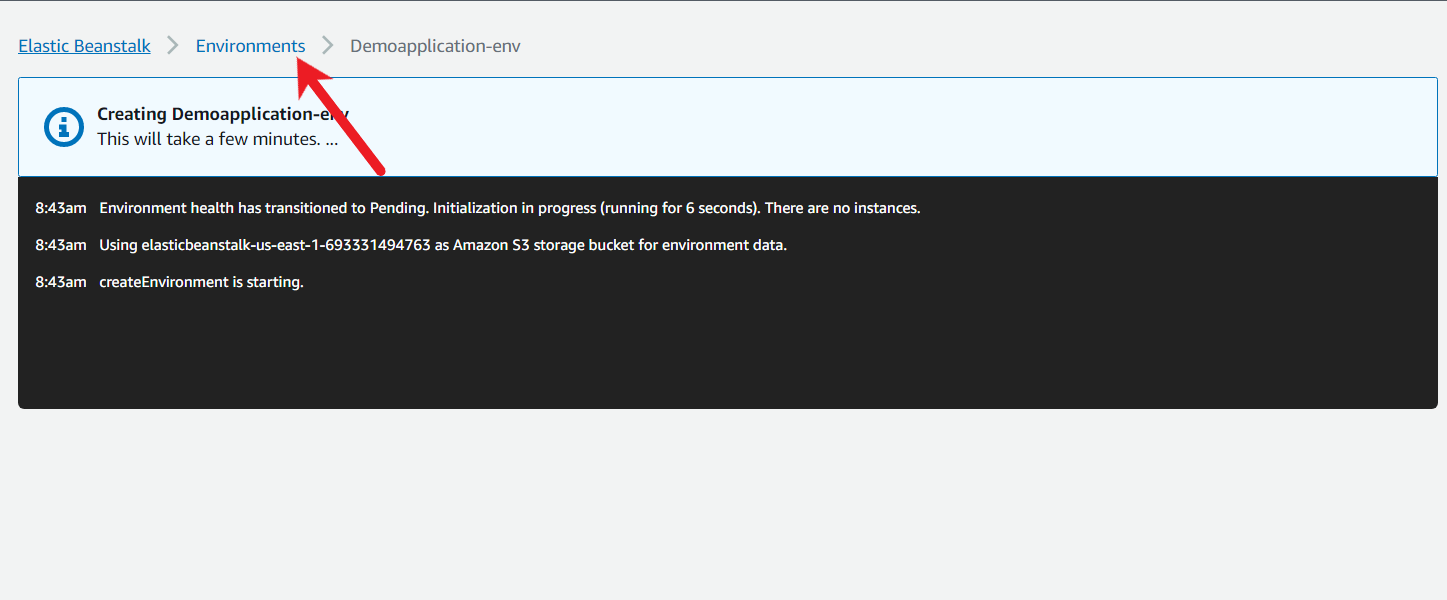
After the file has been uploaded successfully, you will see a "File successfully uploaded" message. Post this click on **Create application**.



1. Elastic Beanstalk will now start with the environment creation process. This creation process can take around 5 minutes to complete.

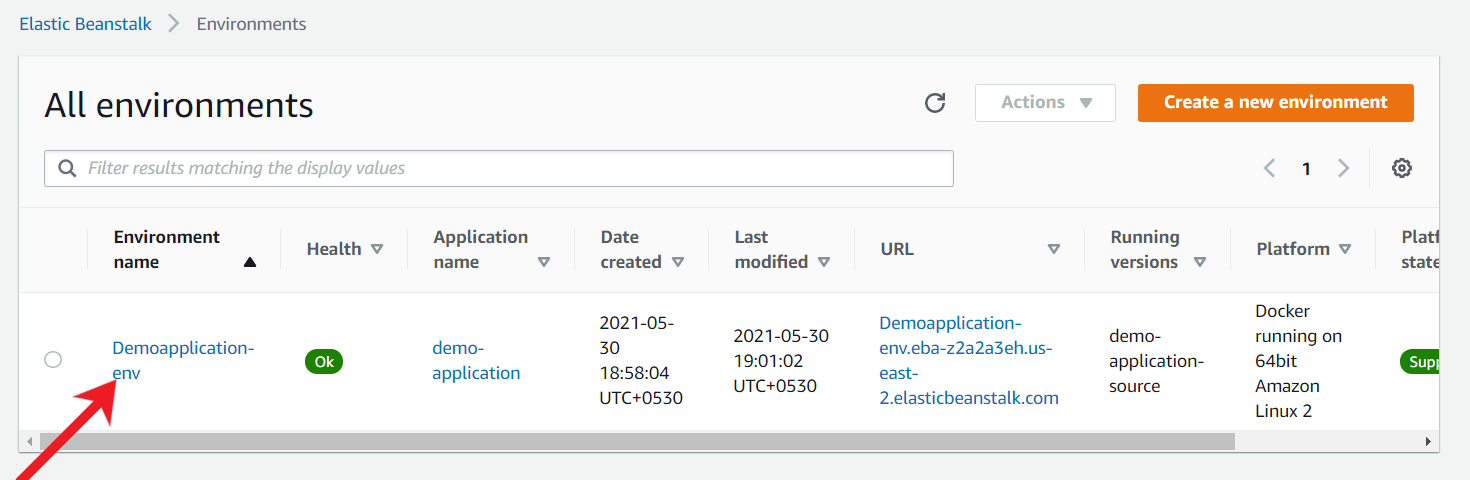


After a few minutes, click on the Environments tab to see the status of the process.

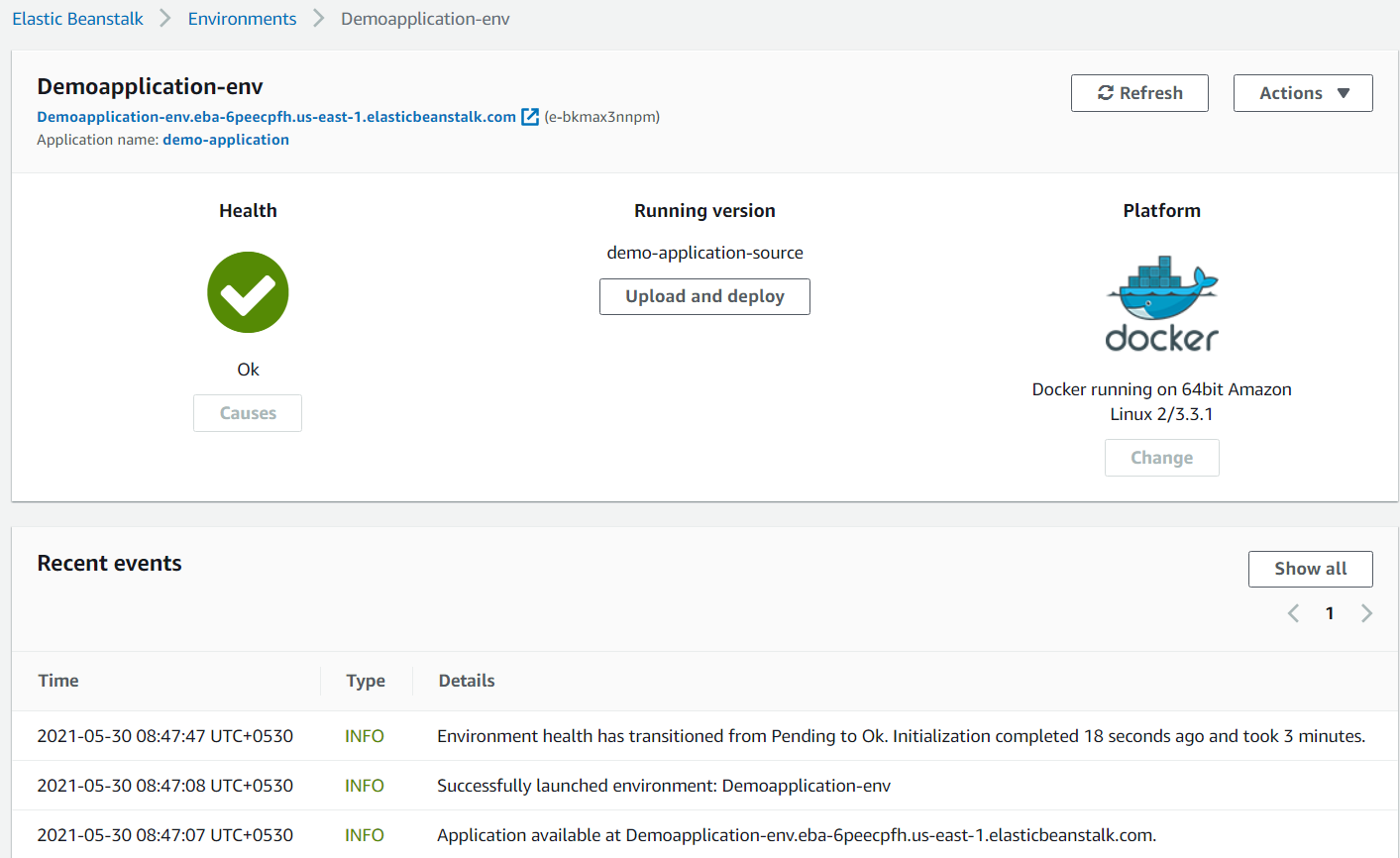


Once the environment has been created, you will see the Health status should be Ok.

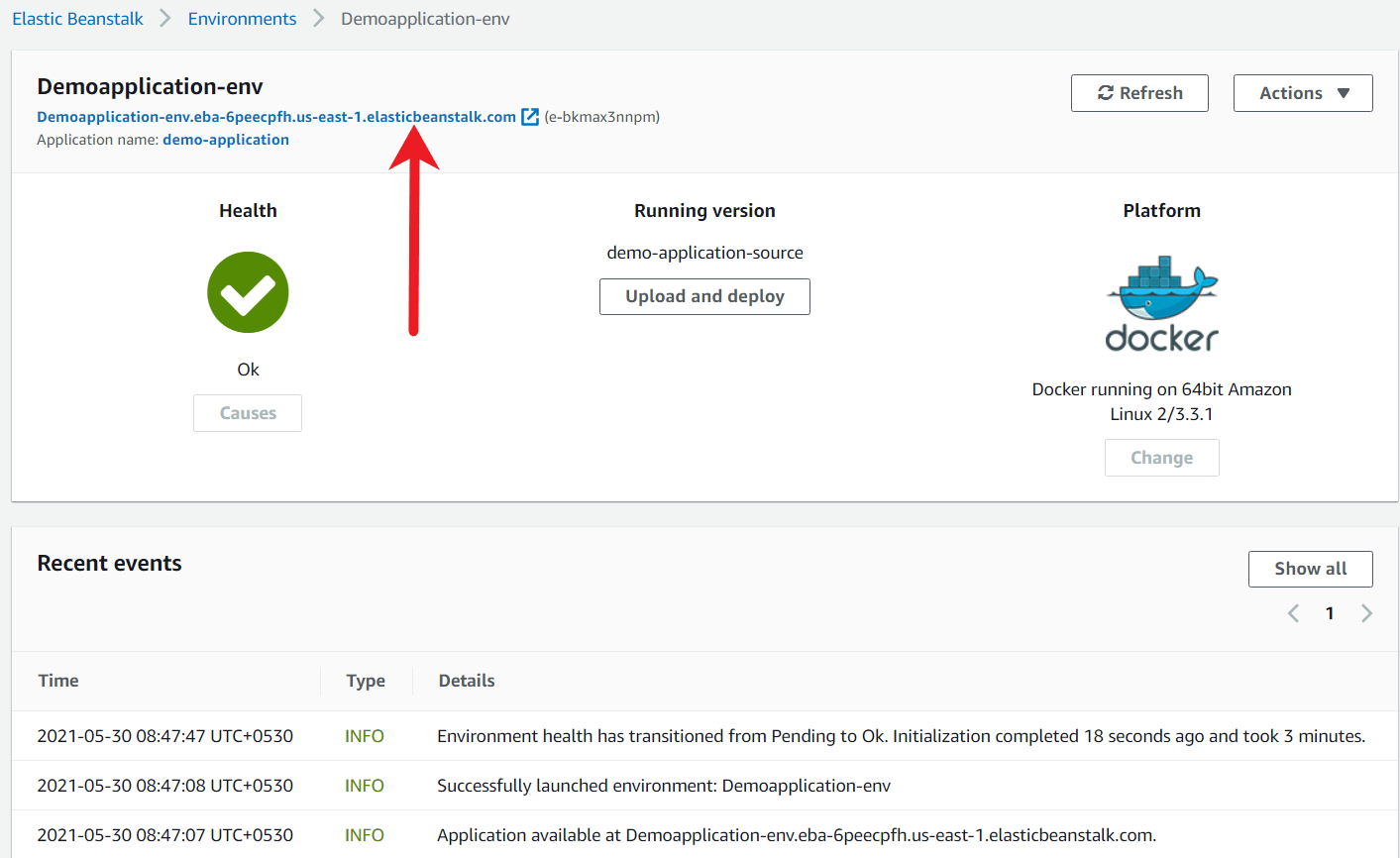
1. Open the created environment by clicking on the environment name.



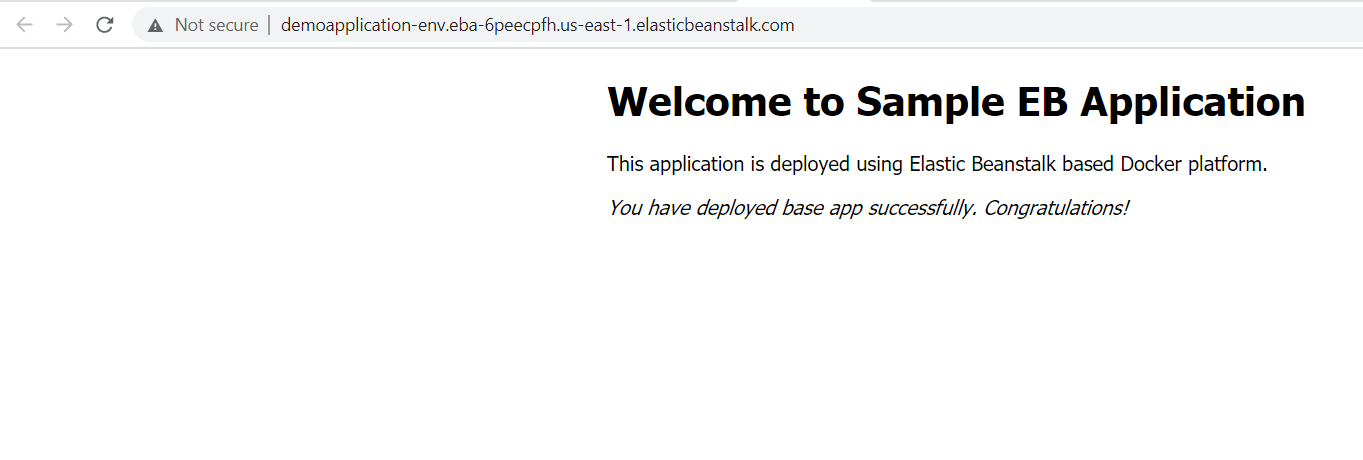
1. In the next screen, you should see all the details associated with your environment, including Health, Platform and the associated events.



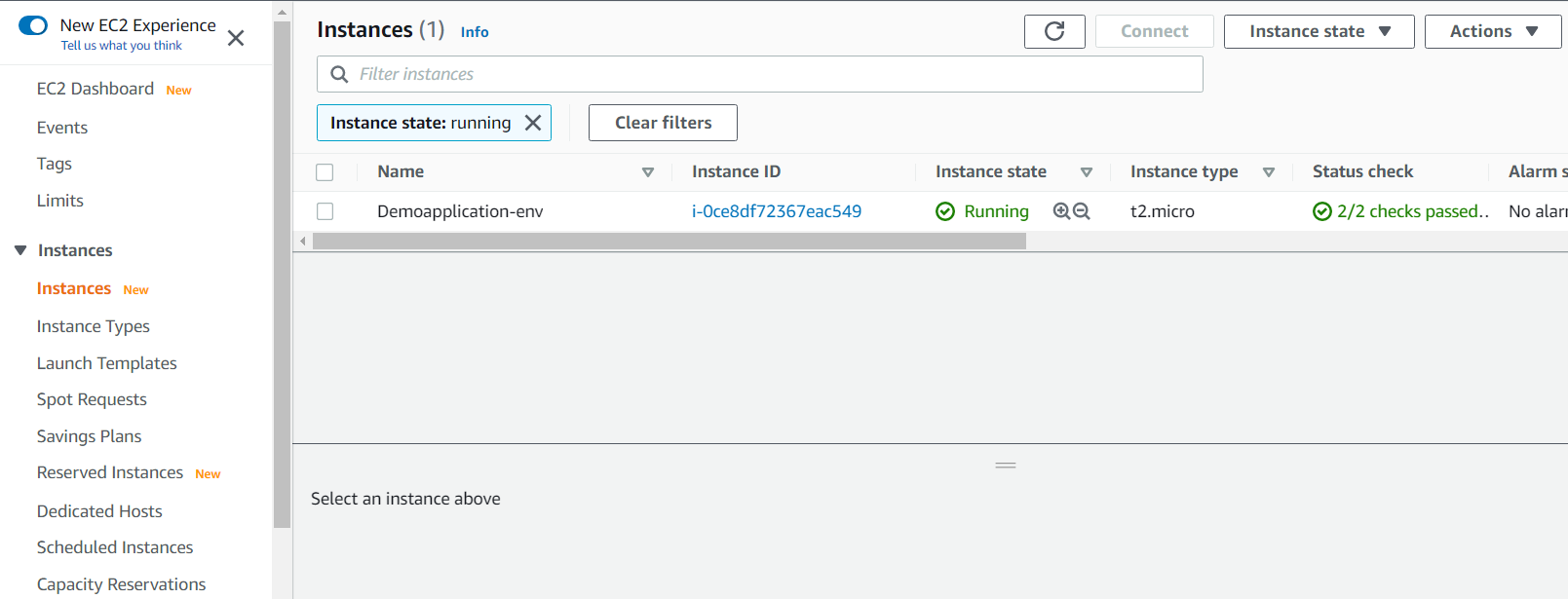
1. To verify if the application is running properly, load the application by clicking on the URL in the main page.



1. If everything is working properly, you should see a webpage with "Welcome to Sample EB Application" as the contents in the browser.



1. You can also verify the underlying resources that are created by the Elastic Beanstalk. Go to the AWS EC2 console, and you should see 1 EC2 instance that is created as part of the Elastic Beanstalk environment that is hosting our application.



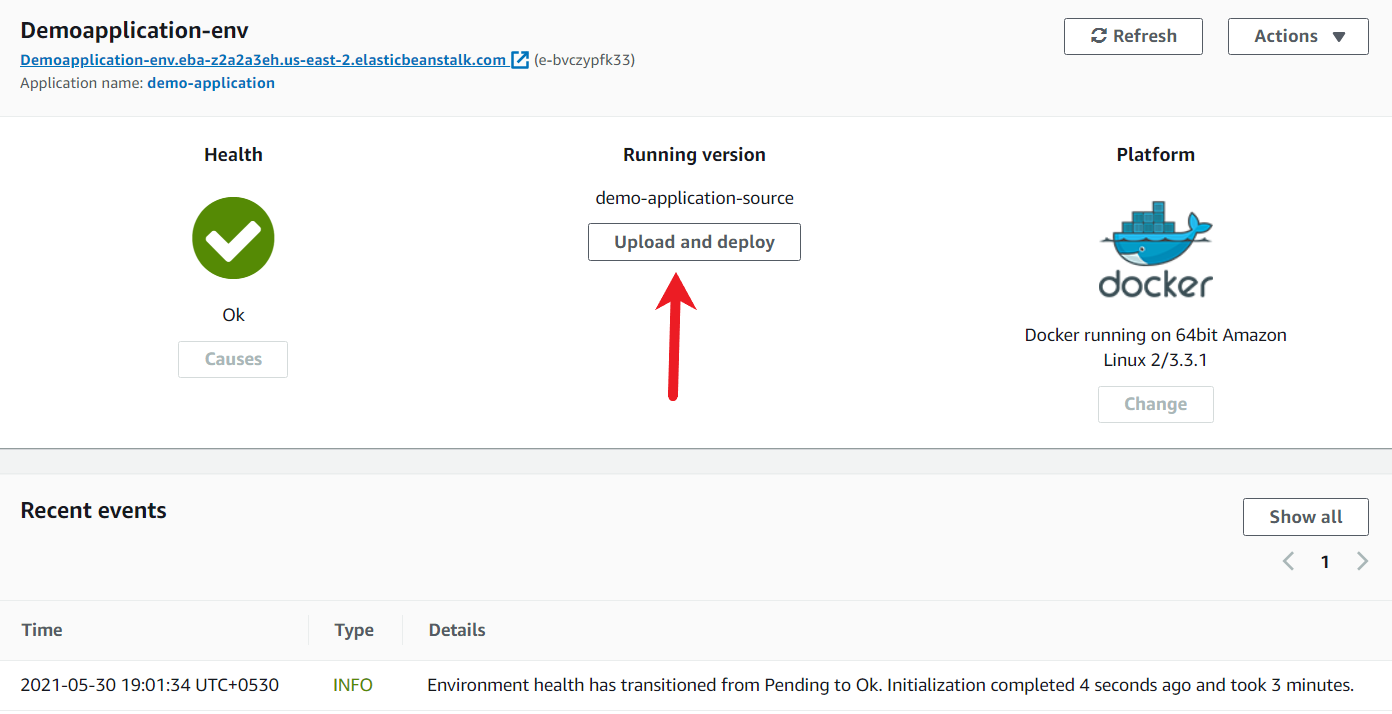
Ref base-eb-app.zip

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features.rolling-version-deploy.html>

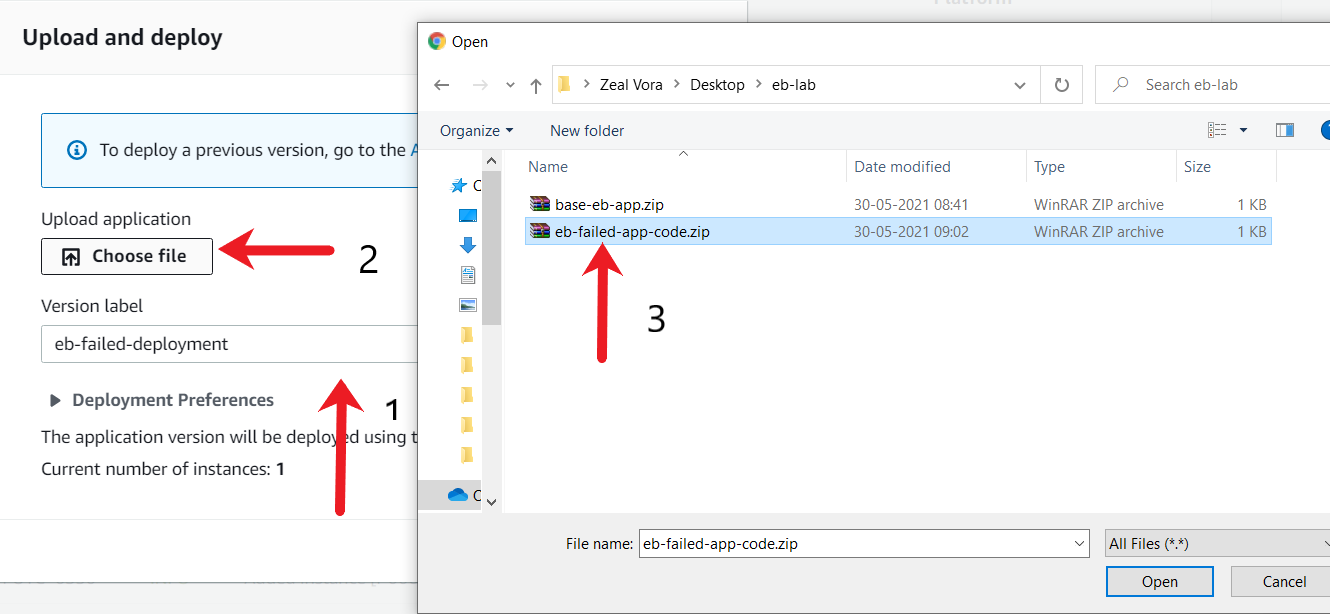
Deploy an Error-ProneApplication

After the basic environment has been created and the application is running successfully, you have to deploy an error-prone application. Document on what happens after the error-prone application is deployed and whether the website continues to run after the failed deployment.

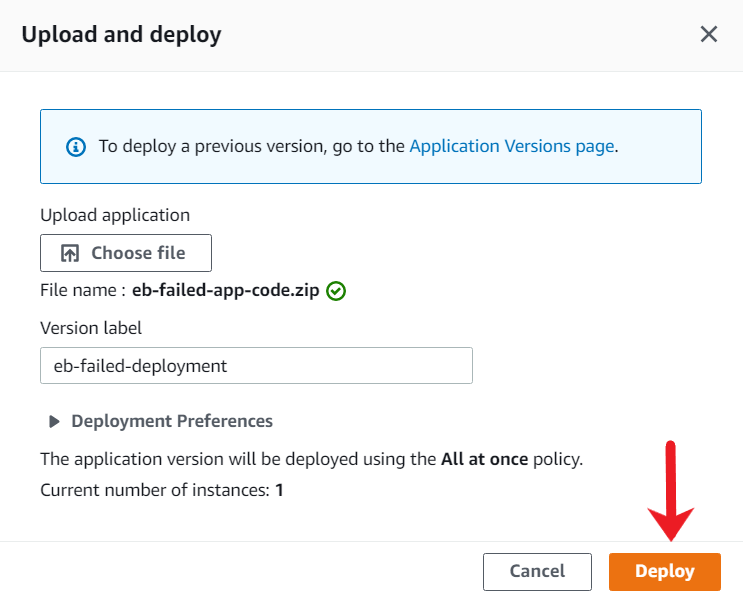
1. From the environment page, under the "Running Version", click on the **Upload and deploy** button.



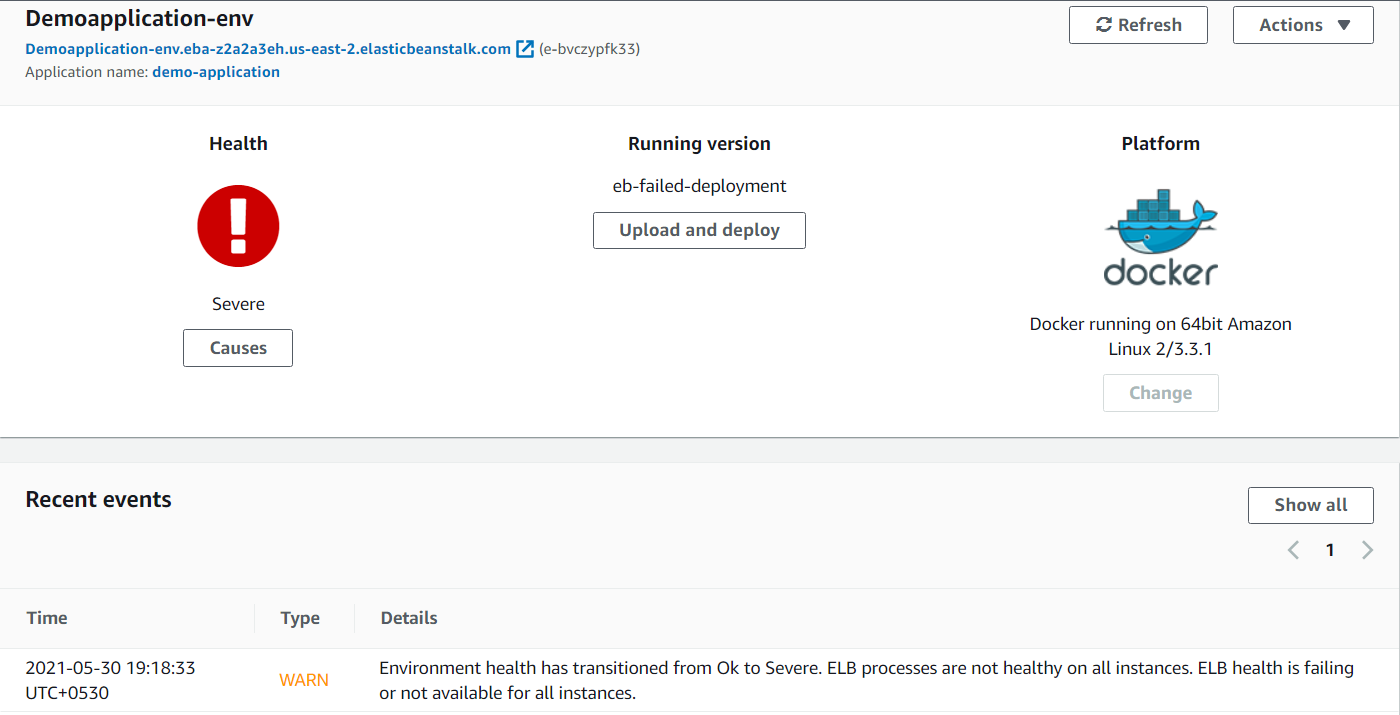
1. Under the new popup, add a version label of eb-failed-deployment and click on **Choose File**button and select the eb-failed-app-code.zip from the local workstation.



1. Once the zip archive has been uploaded, click on "**Deploy**".



1. After the deployment completes, the environment's health will turn from Green (Healthy) to Severe similar to the following screenshot. It can take around five minutes for the change in health. Click on the "Refresh" button to get the latest state.



1. Once the application update has been completed, click on your application URL. You should see a 502 Bad Gateway message.



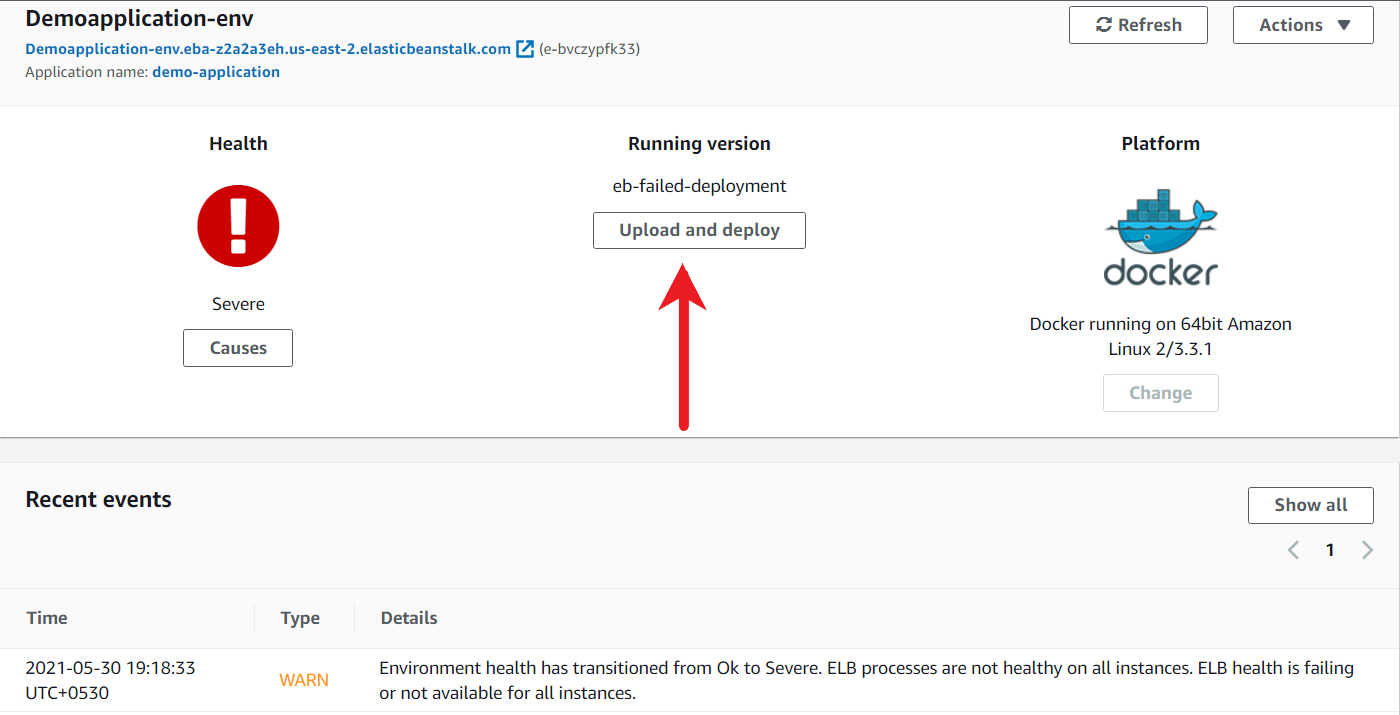
We can conclude that whenever a default Elastic Beanstalk deployment setting is used, and if there is a failed application deployment, then the production server can go down, resulting in a 502 bad gateway for all the customers.

Ref eb-failed-app-code.zip

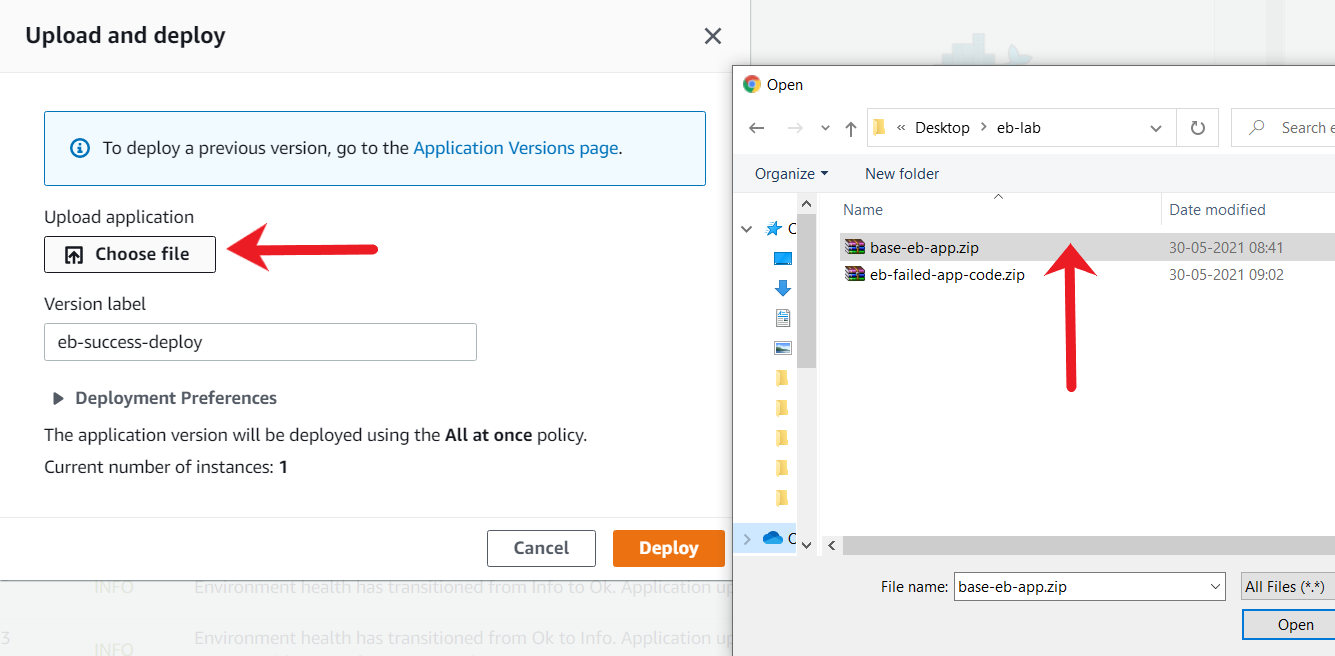
Configure the Deployment Policy

Based on the requirements, you have to configure an appropriate deployment policy for new updates that are made to the application. This deployment policy should be implemented so that during deployment, the existing production capacity of servers is not altered, and failed deployment would not result in the primary application going down. Before making any changes, make sure you re-deploy the working application.

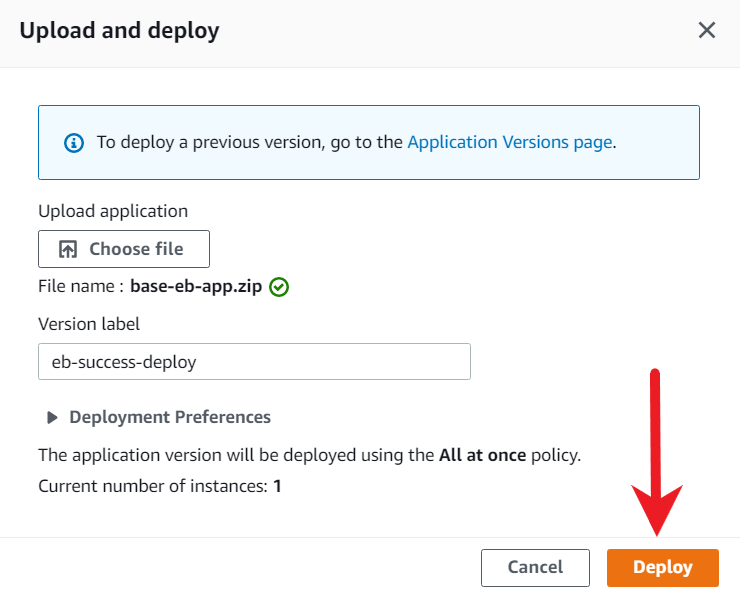
1. From the environment page, under the "Running Version", click on the **Upload and deploy** button.



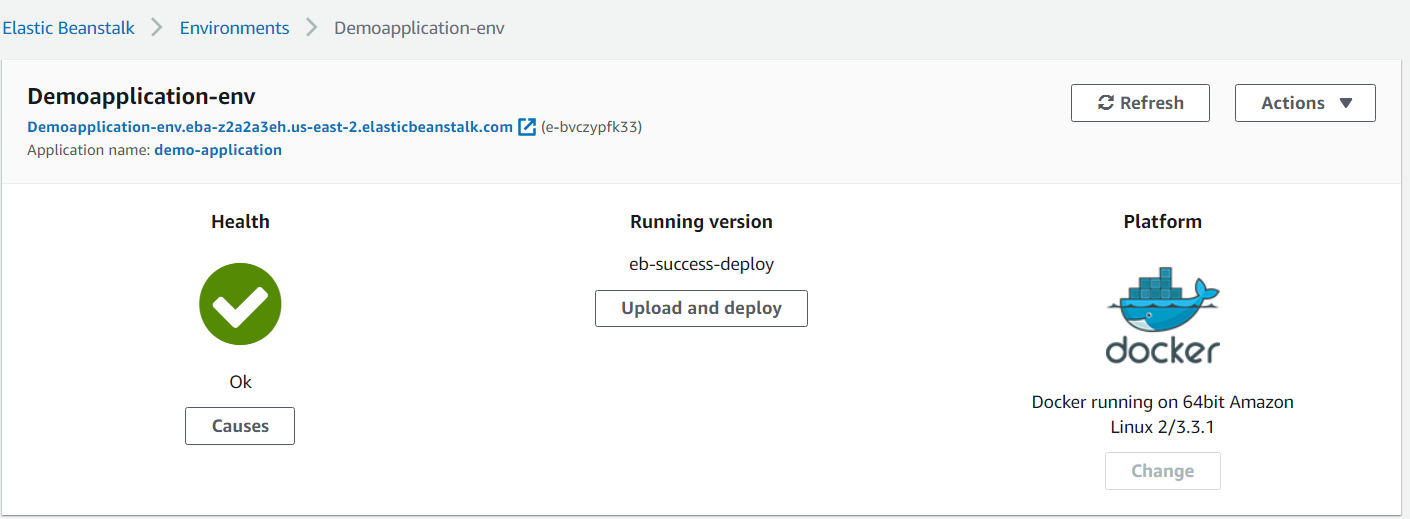
1. Under the new popup, add a version label of eb-failed-deployment and click on **Choose File**button and select the base-eb-app.zip from the local workstation.



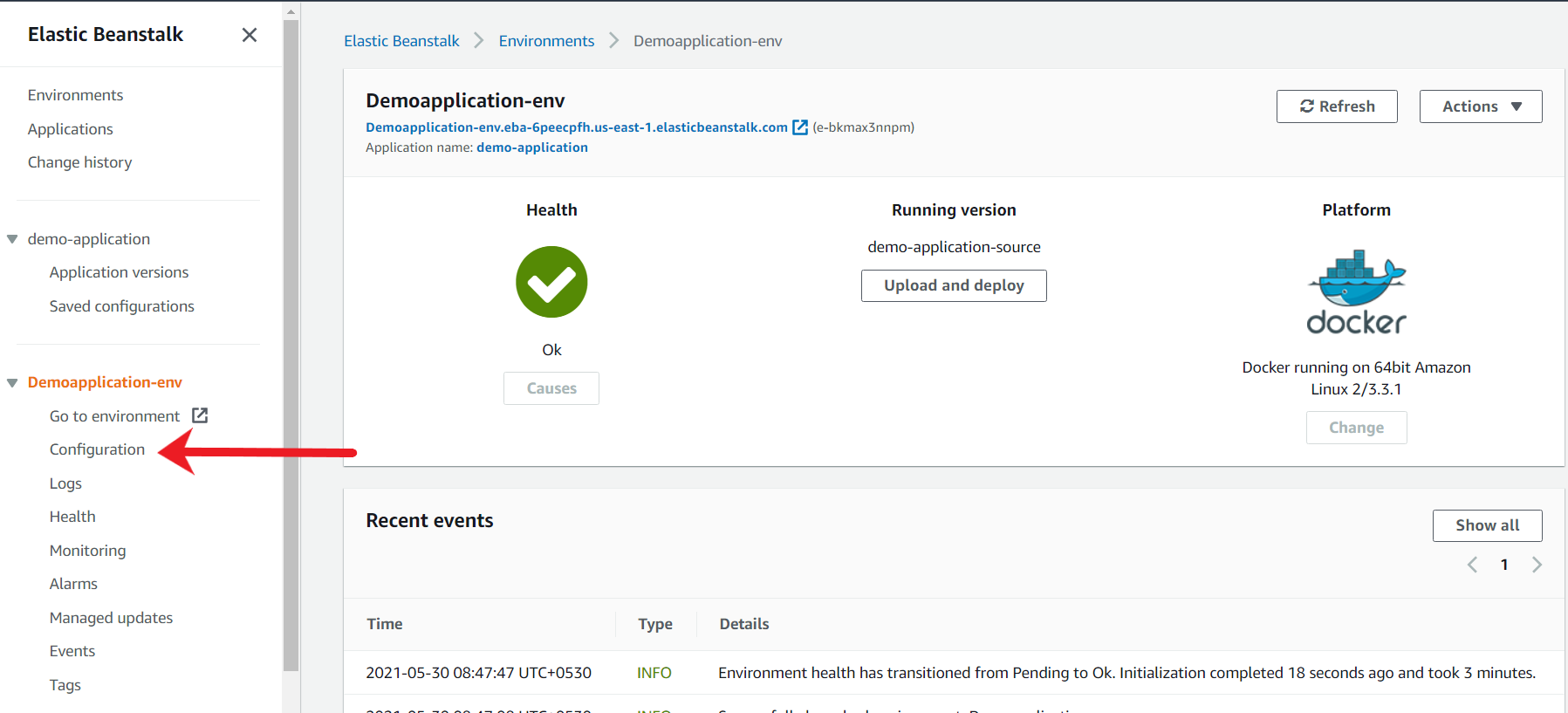
1. Once the zip archive has been uploaded, click on "**Deploy**".



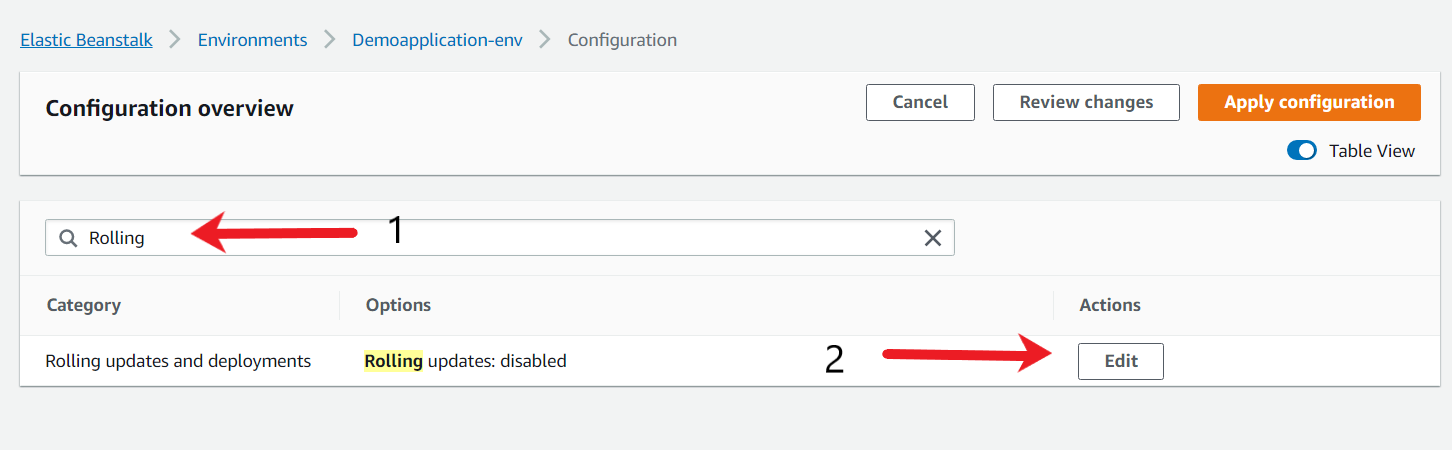
1. After the deployment completes, the environment's health will turn from Green.



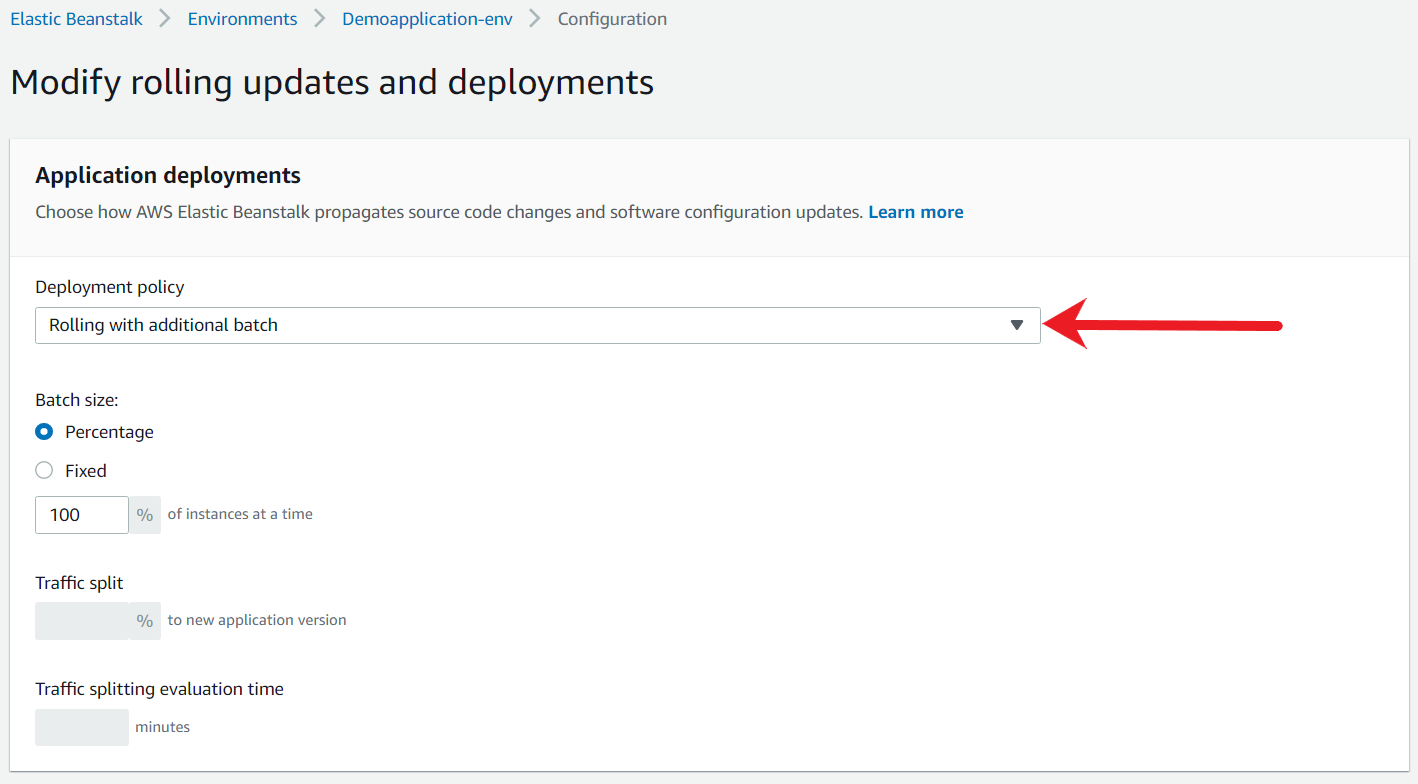
1. To change the deployment setting, click on the **Configuration**tab.



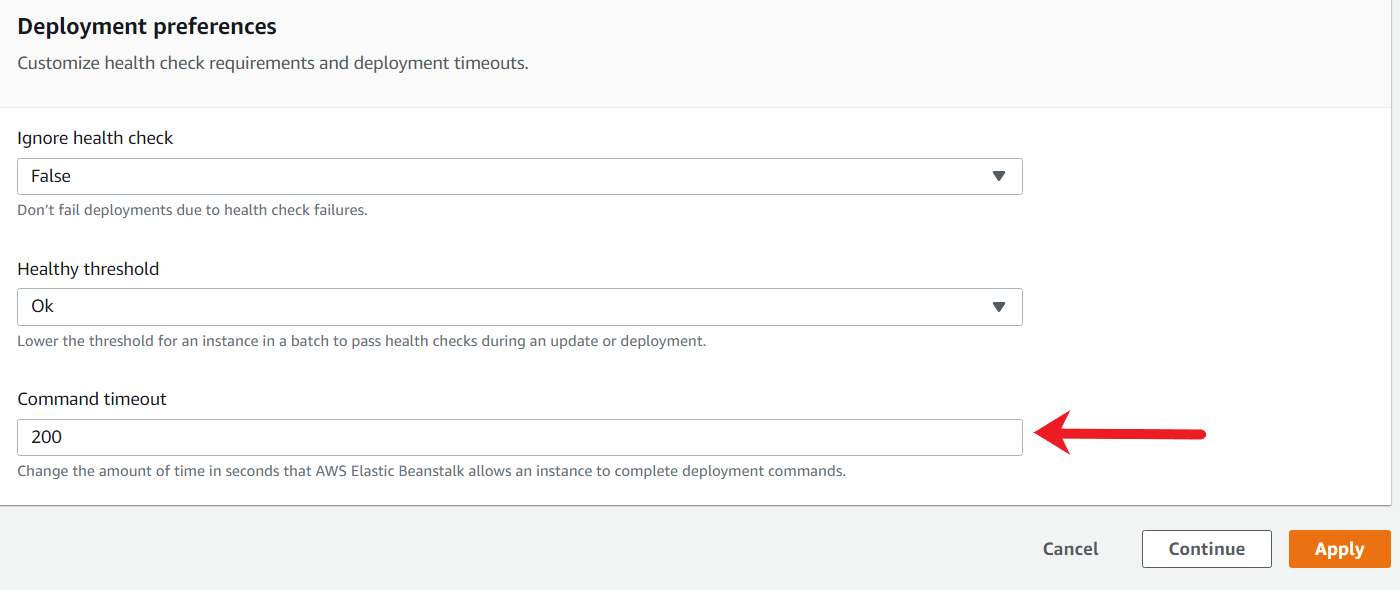
1. Type "Rolling" in the search window. For the category of "Rolling updates and deployments" click on Edit.



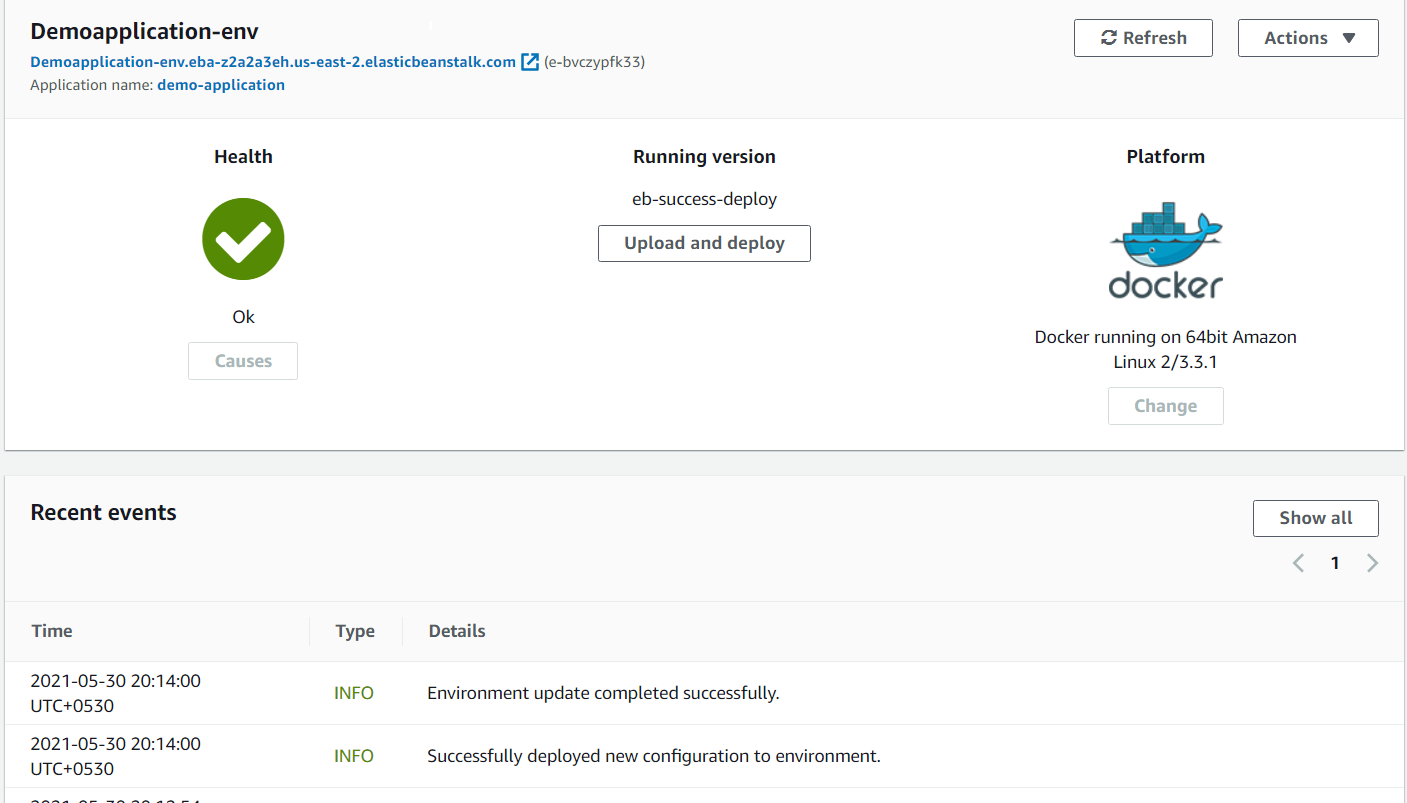
1. Under the Deployment Policy, choose "**Rolling with additional batch**".



1. Additionally, scroll to the bottom of the page, under the "Deployment Preference" section, set the command timeout to 200 instead of 600. This will allow us to quickly test the deployments. Once done, click on "Apply".



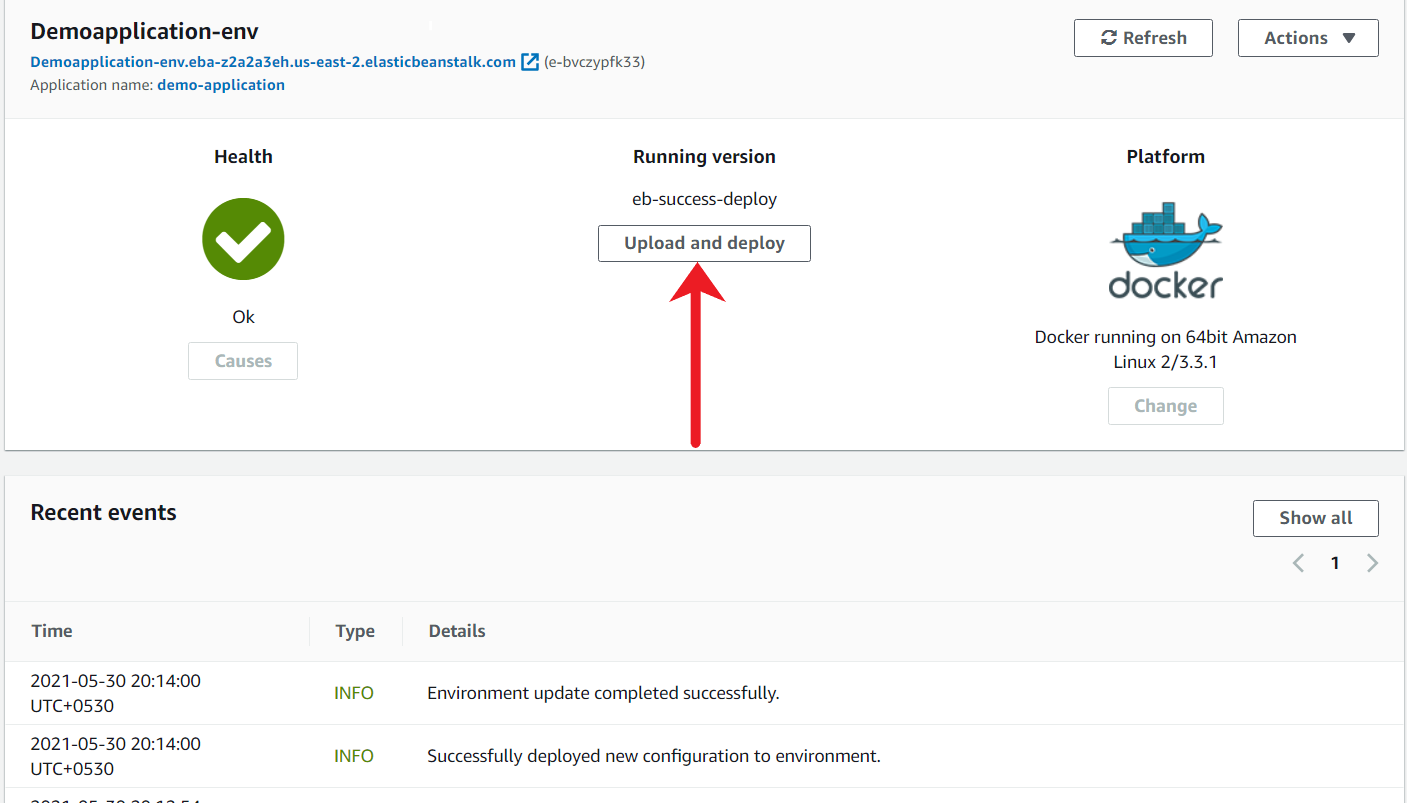
1. After the apply is successful, you will be redirected to the environment page. Verify if the health is Green.



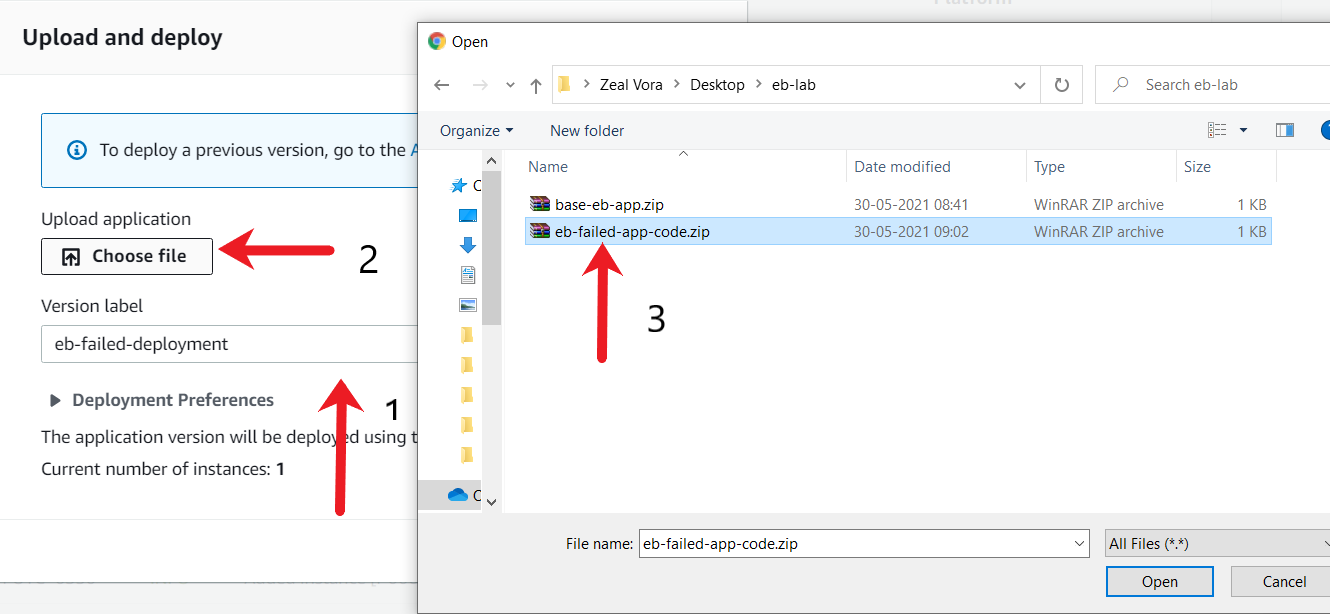
Test the New Deployment Policy

After you have configured a new deployment policy, simulate the failed deployment to verify how the new deployment policy handles failures. Upload the failed application version yet again to complete the verification.

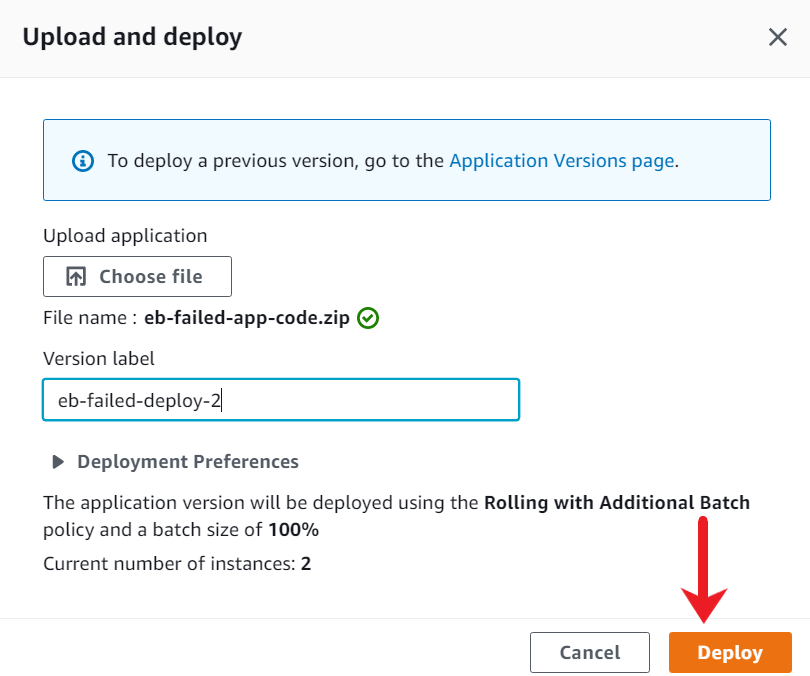
1. From the environment page, under the "Running Version", click on the **Upload and deploy** button.



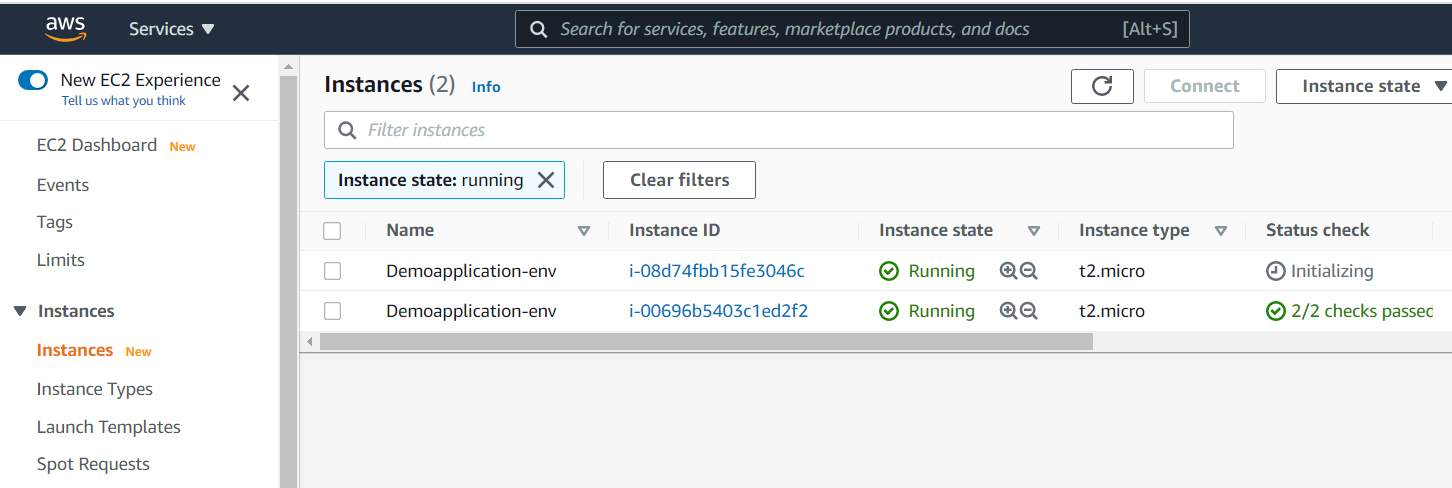
1. Under the new popup, add a version label and click on **Choose File**button and select the eb-failed-app-code.zip from the local workstation.



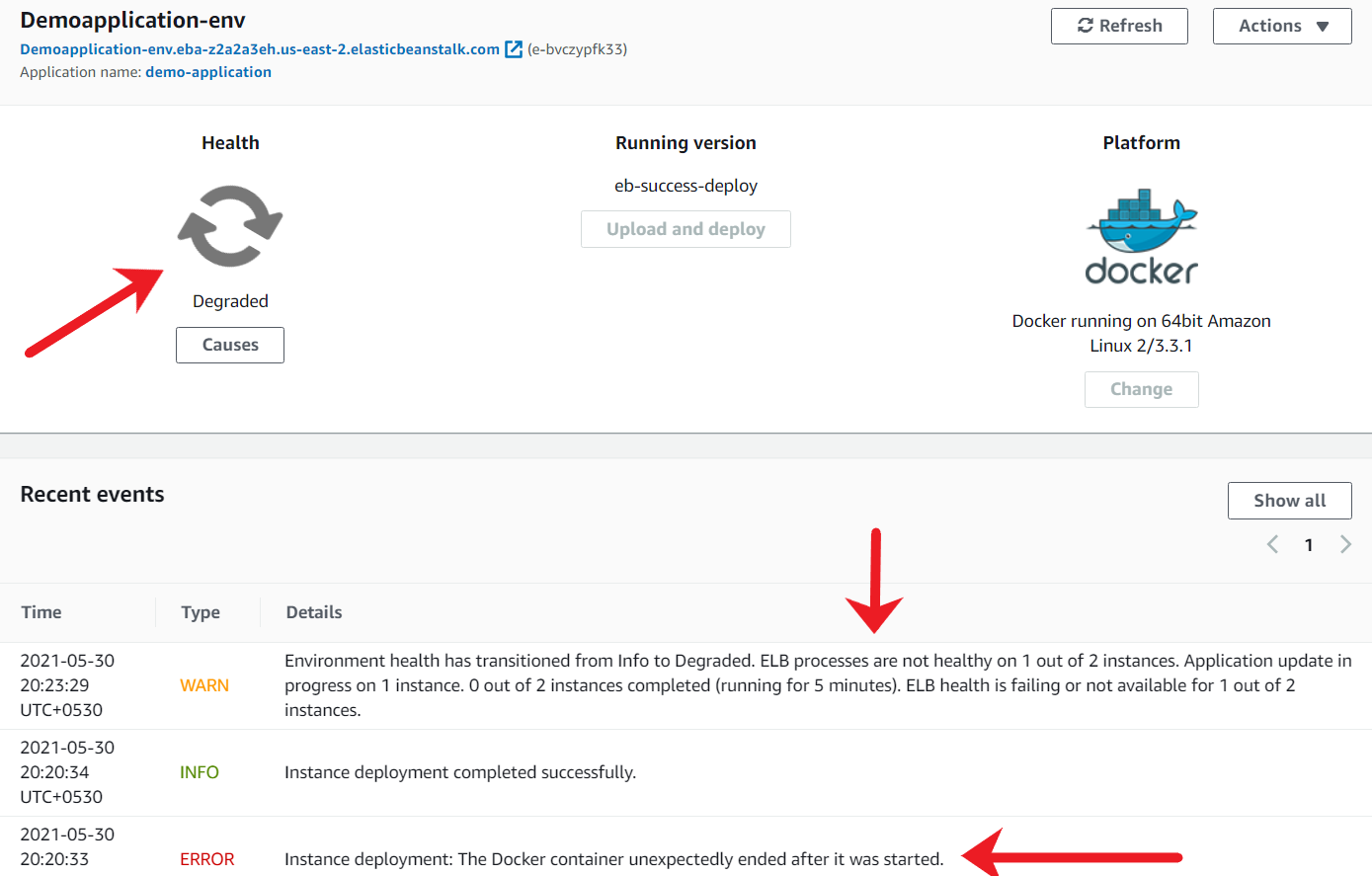
1. Add an appropriate version label. Once the zip archive has been uploaded, click on "**Deploy**".



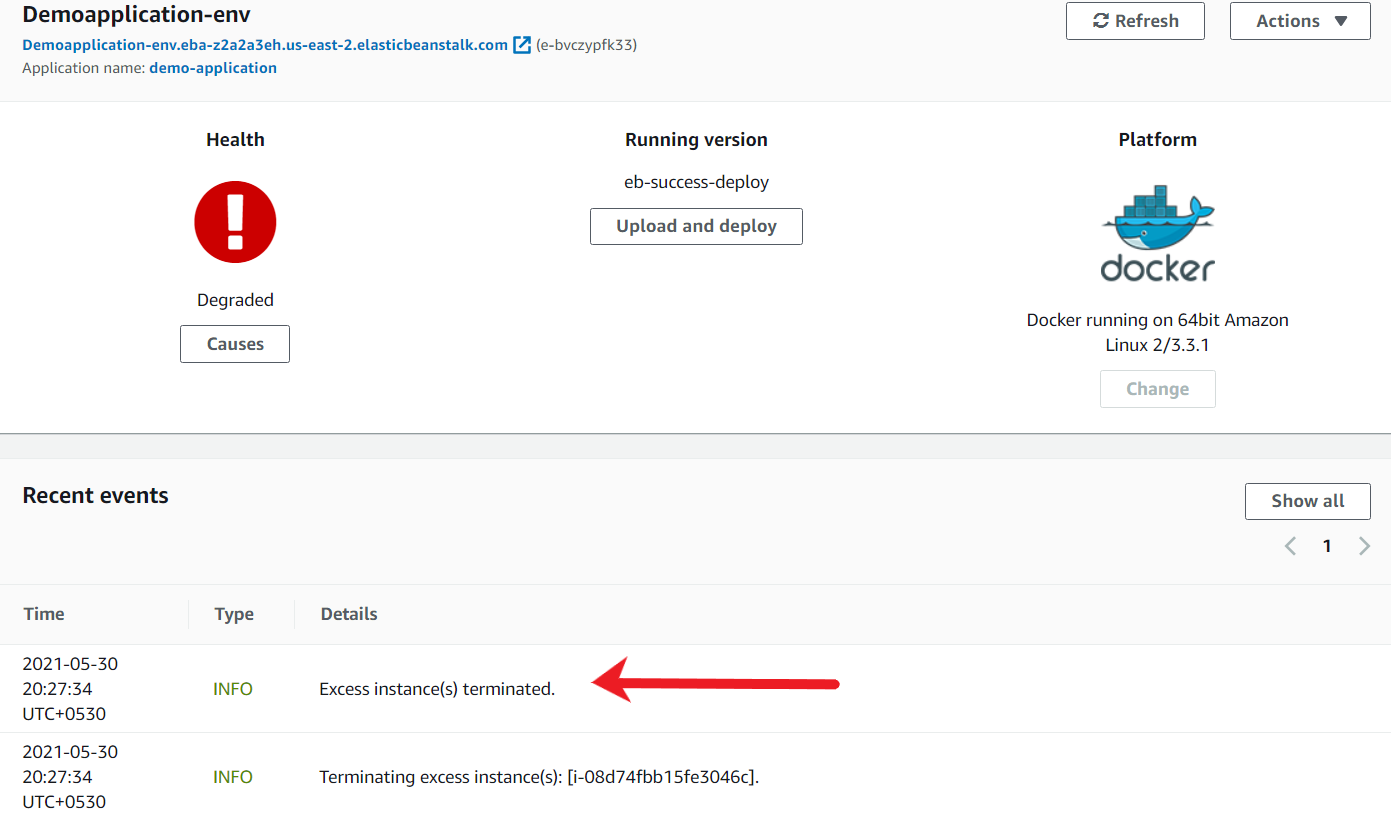
1. Go to the EC2 console, and verify if an additional EC2 instance is being created. This new instance is where the new application will be deployed based on the new deployment policy.



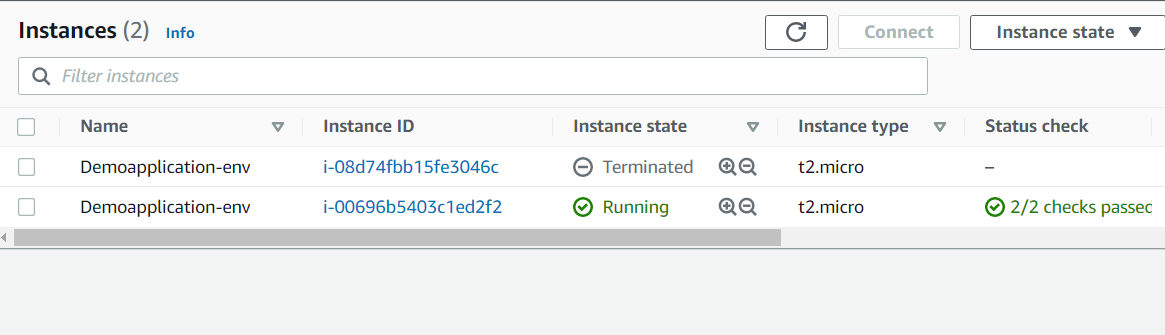
1. In the Elastic Beanstalk console, after a few minutes, you should see the environment status to be degraded. Under the recent events, you should see multiple ERROR and WARN based events.



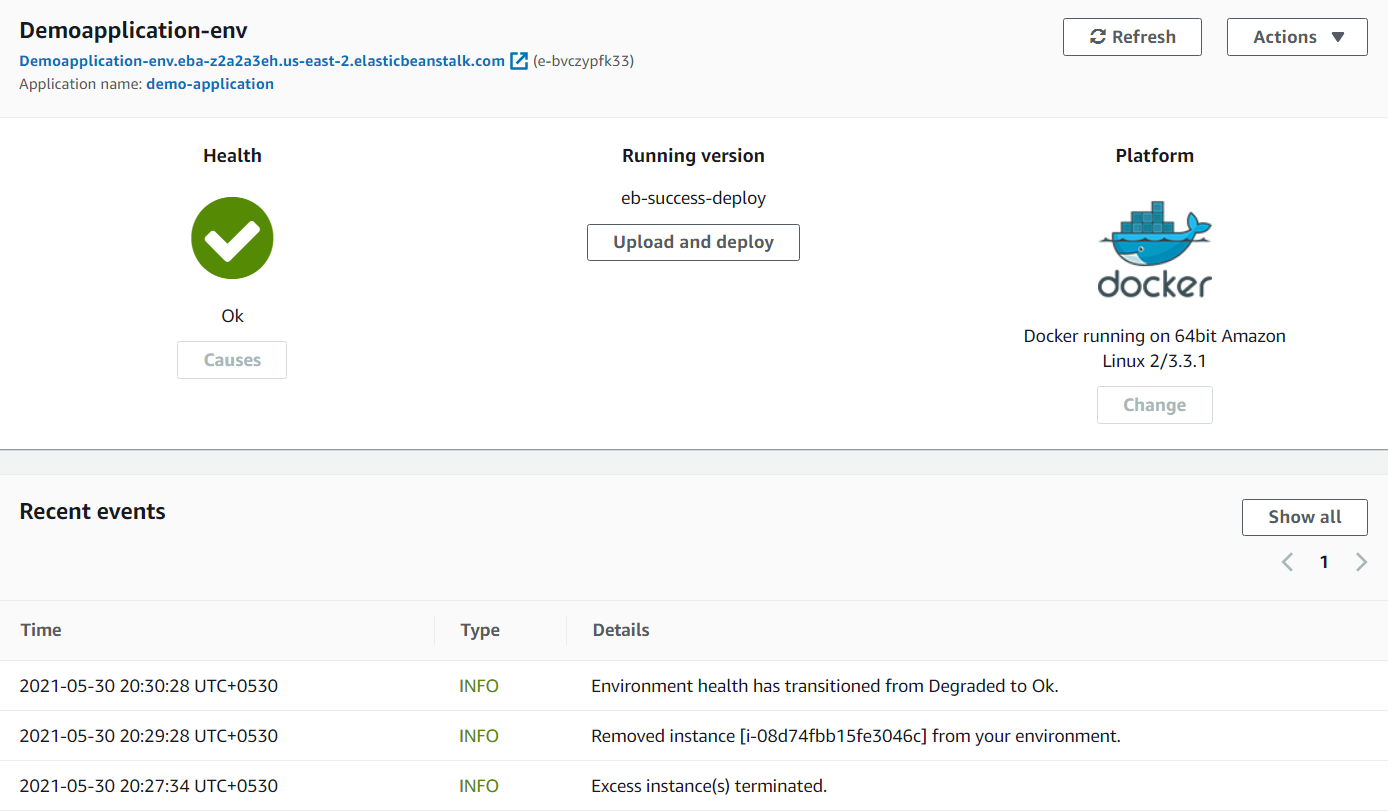
After a while, the final environment status would be "Degraded" state, and under the recent events, you should see the newer instance is terminated.



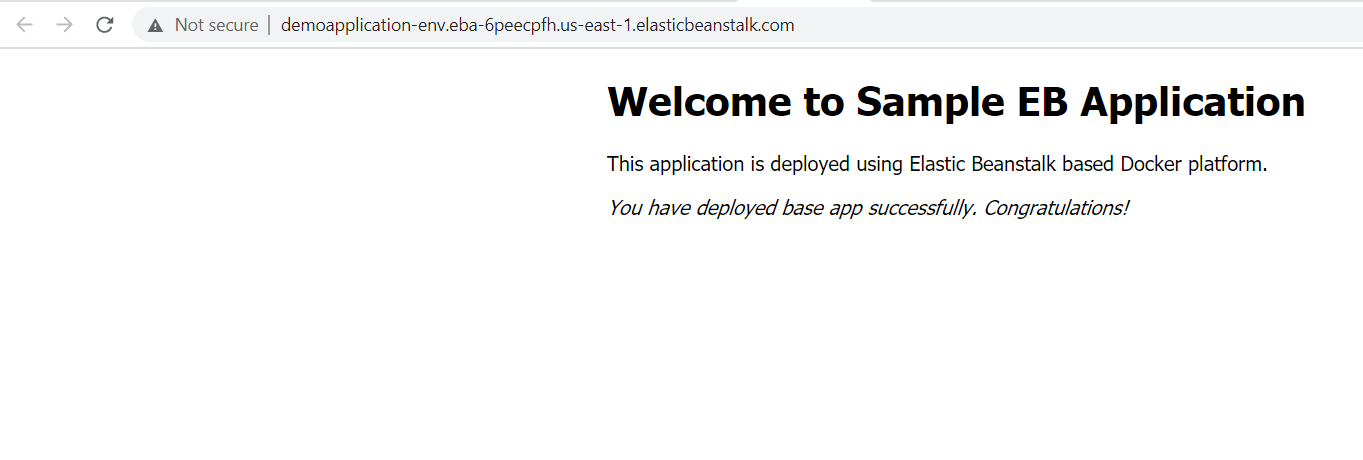
1. From the EC2 console, verify if the additional EC2 instance is terminated.



1. After a few minutes, the Elastic Beanstalk environment should automatically turn itself as Healthy yet again, and the failed deployment should no longer exist.



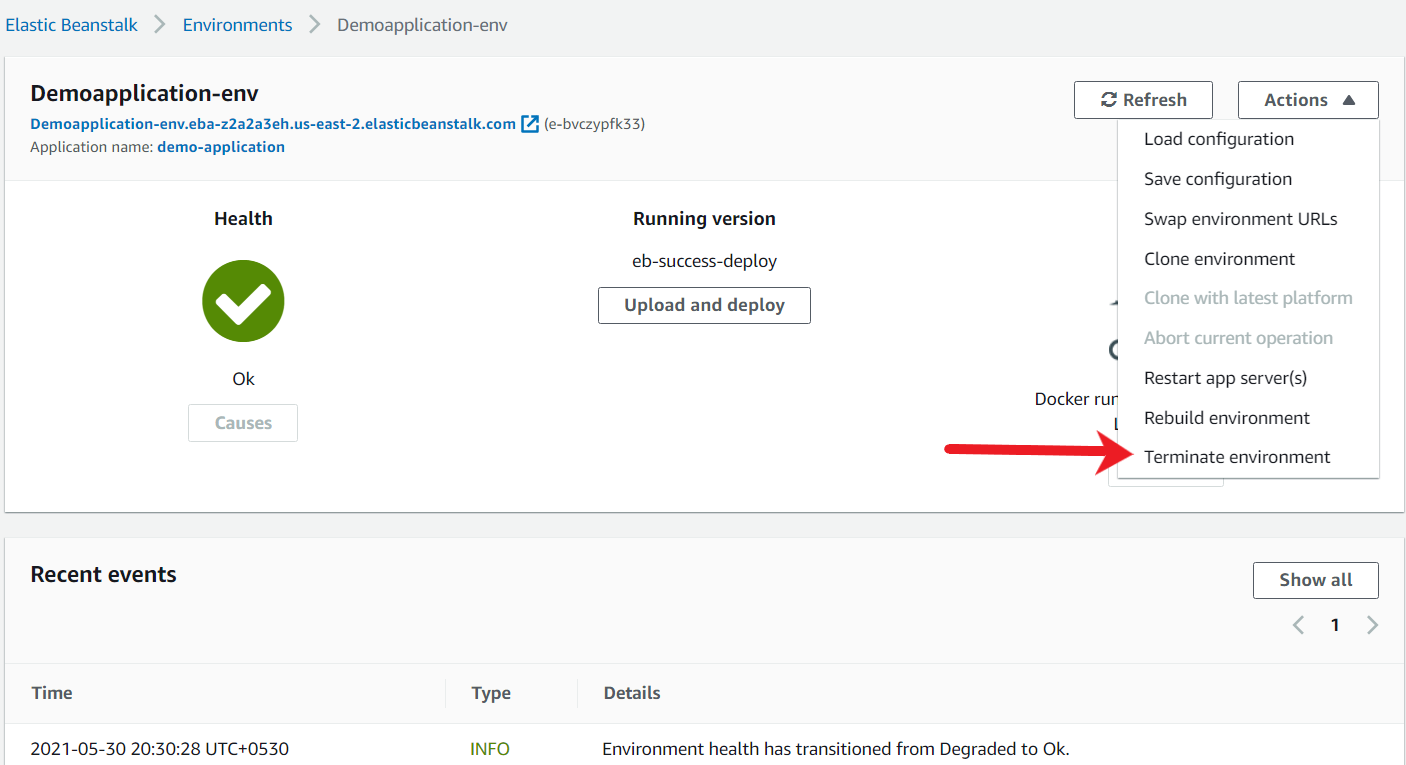
1. Once the environment update is green, click on your application URL. You should see the application up and running.



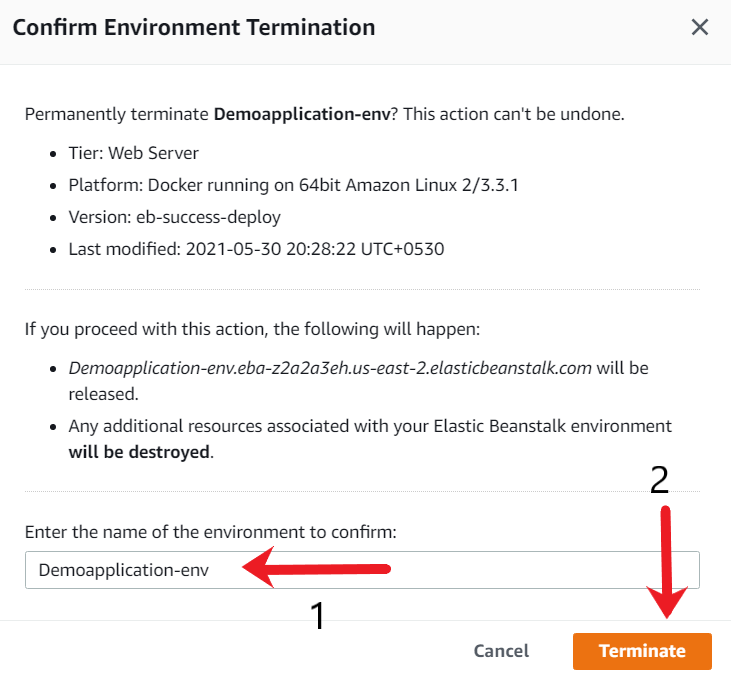
Terminate the enviroment

As you have successfully demonstrated a proof of concept for the deployment policy, clean up the AWS resources at this time. You will need to delete all the resources created for your proof of concept in the environment.

1. To terminate the Elastic Beanstalk Environment, select your appropriate environment, click on Actions, and choose Terminate environment.



1. Use the on-screen dialog box to enter the environment name and click on **Terminate**.



1. You can also verify the termination by going back to the EC2 console, and you will see that all the instances are terminated.

