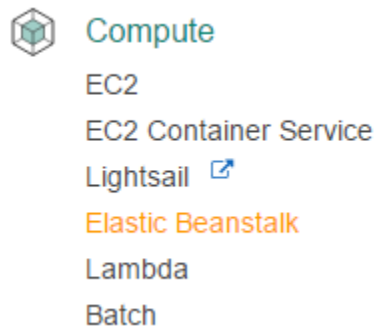
This Lab covers:
- How to build an Elastic Beanstalk Application container and environments
- The efficient way to perform a rolling deployment on underlying EC2 resources
- Blue-Green deployment with Elastic Beanstalk for zero-downtime launches
- How to mitigate cost by validating cleanup after immutable or blue-green deploys

During this lab, you will walk deployment of a new load balanced, auto scaling Todo application into an AWS VPC, upgrade the code of your app to version 2 using low-friction and rapid rolling deployment, construct a secondary deployment stack, upgrade the secondary for use as the primary, swap the environments with no downtime, and clean up the older versions of code to save money on resources.

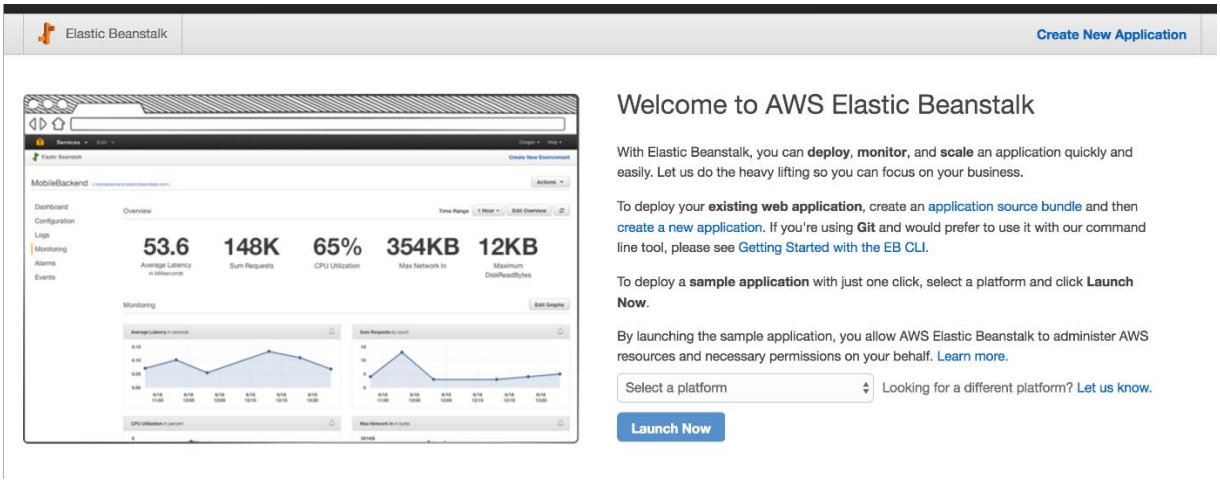
Step 1. Create an Elastic Beanstalk App & Environment

Since we will be performing both a rolling and blue-green deploy in this lab, it makes sense to use the best tool AWS makes available for the job. For controlled deployments and efficient deployment services of code on EC2 instances, Elastic Beanstalks provides a superior interaction model and developer tools experience.

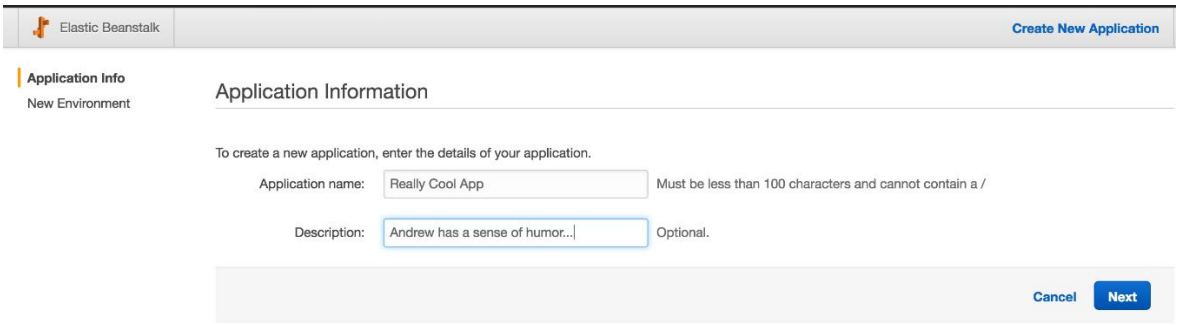
First, we navigate to the AWS Elastic Beanstalk Console, by clicking on the **Services** tab in the top-right of the Console, hovering over the **Compute** Section in the tray, then clicking on the **Elastic Beanstalk** section...



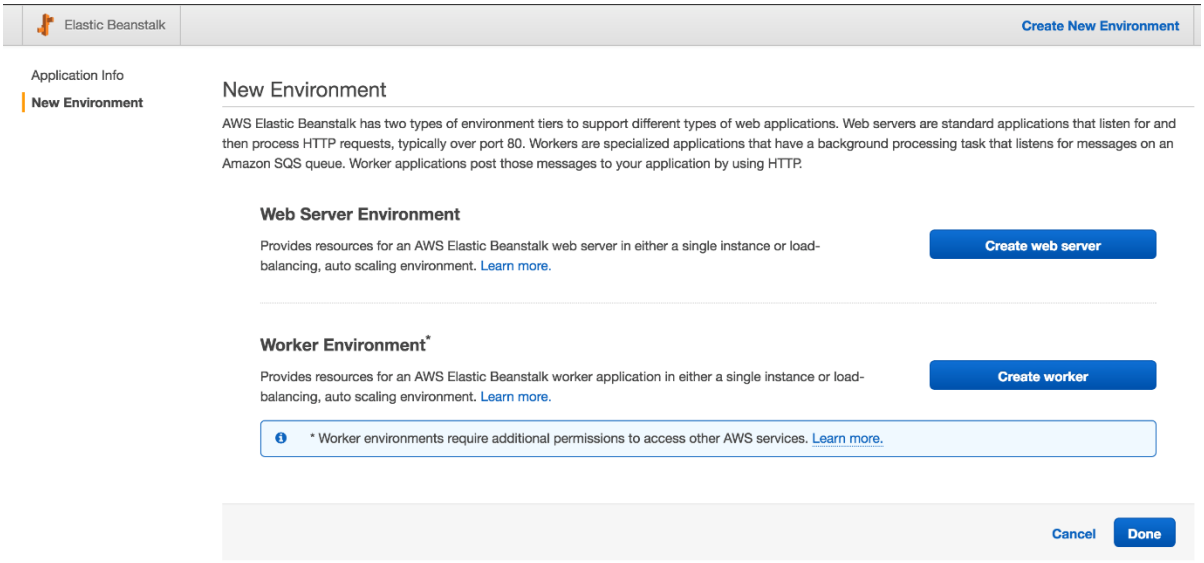
Once inside the Elastic Beanstalk Console, because the account the Lab uses has never used Elastic Beanstalk before, you will be presented with a welcome message. The most obvious button in the center of the screen is actually **NOT** the one we want. That one launches a simplified demo application immediately. Click on the **Create Application** link in the top-right of the Console view.



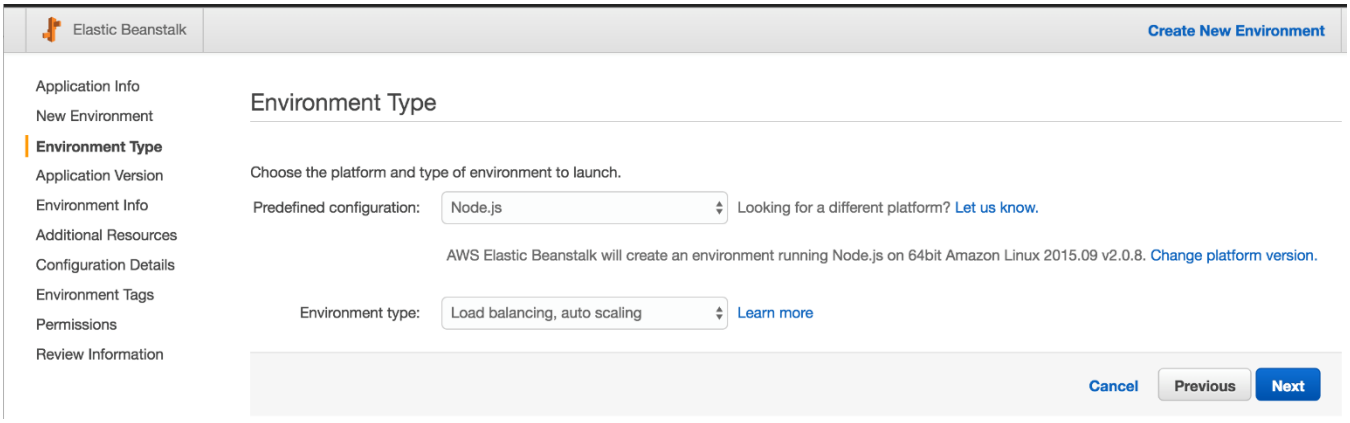
You will be immediately presented with options. The first of which are two text blanks for **Application Name** and **Description**. These are vanity names, and do not really affect anything, so you can pick whatever you like that AWS accepts. Spaces are allowed in both the name and description. Click the blue **Next** button in the bottom right of the view.



Next, you are presented with some options for the Environment Type you want to launch. We will be launching a **Web Server Environment**, so click on the large blue button that says **Create Web Server**.



Then, you will be presented with two drop-downs, the top of which allows you to choose the platform or programming language in which your application will be authored. The code this Lab provides you is all **node.js**, so in the top dropdown, you should select this value. The second dropdown allows you to select either **Load Balancing, Auto Scaling** or **Single Instance** for the **Environment Type**. Fortunately for us, once you select **node.js** in the top dropdown, **Load Balancing, Auto Scaling** will be automatically selected - this is the value we want. Once you have configured these values and confirmed they match the image below, hit the blue **Next** button in the bottom right of the view.

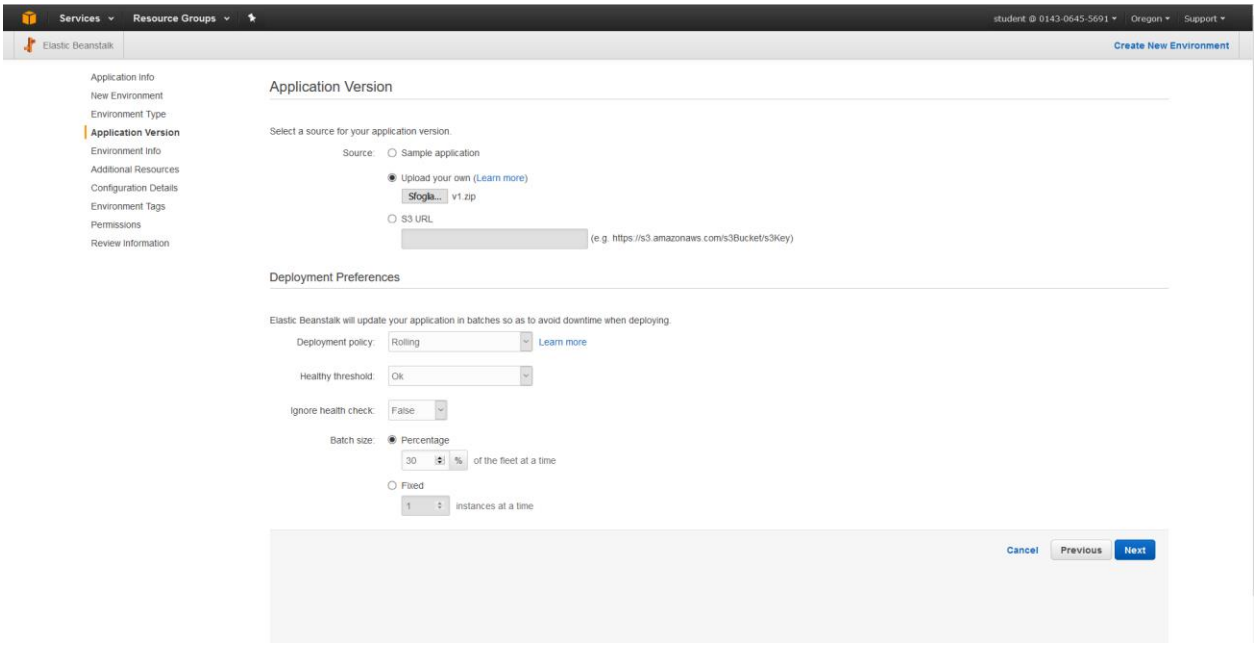


In the next screen, you are presented with a number of options relating to code deployment. The only one we will configure is the **Source** field. Please download this zip file, which contains the

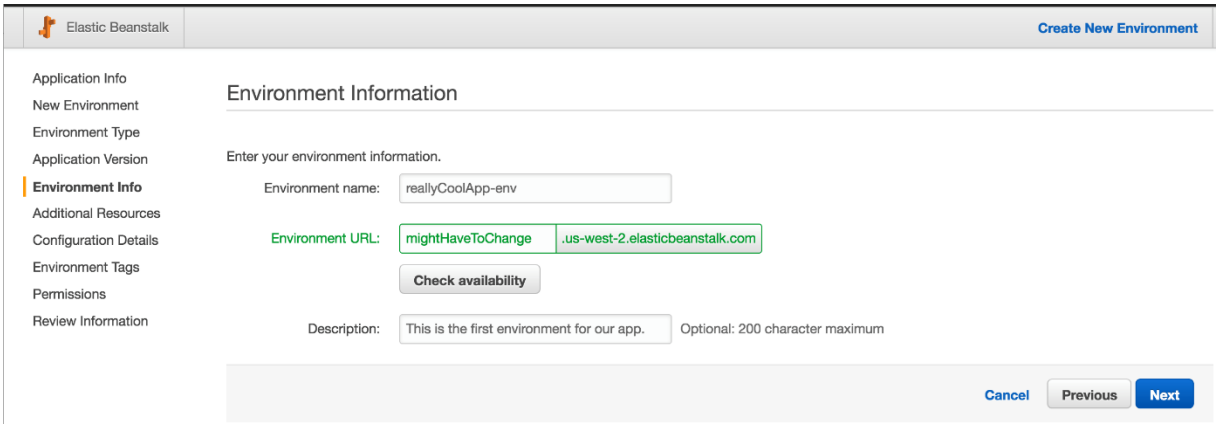
application we will use, then select it for upload using the **Choose File** button associated with the **Upload Your Own** option in the **Source** field under the **Application Version** heading.

We will ignore the **Deployment Preferences** for now - these controls do not matter very much for the lab. These controls configure how quickly and in what size batches the later **Rolling Updates** we'll configure will execute.

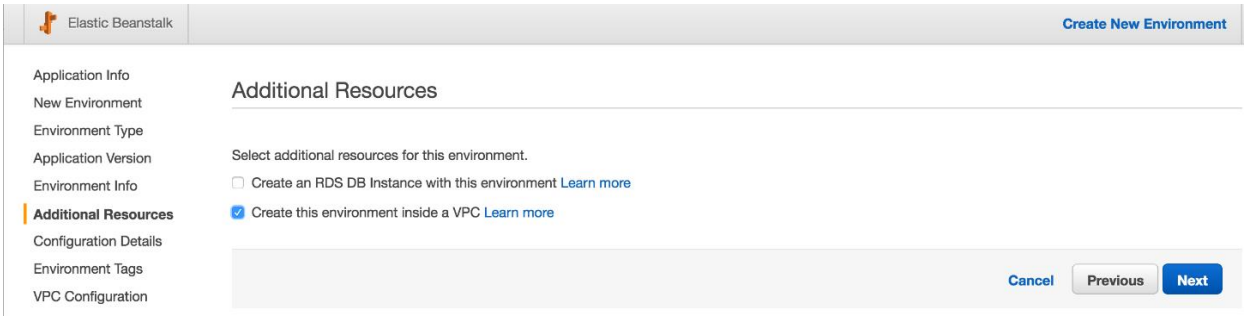
Once you have uploaded the **Source** bundle, please click the **Next** button in the bottom-right hand corner of the view to continue.



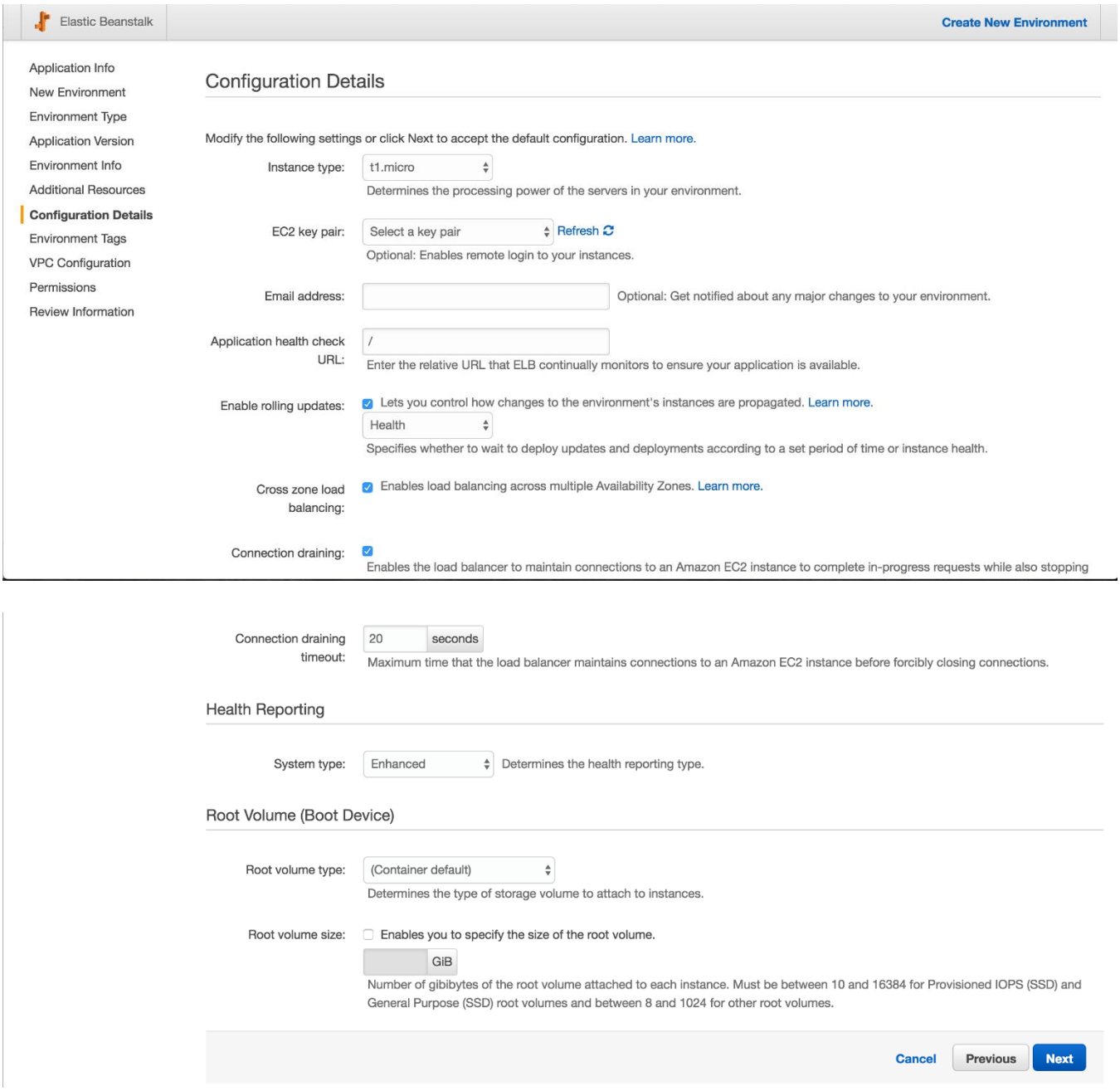
You should now be on a view with the heading **Environment Information**. This setting will play a key role later. Note that AWS attempts to auto-populate your **Environment Name** and **Environment URL** fields for you. This will sometimes not work, because the URLs operate on a subdomain basis - so all users on all AWS regions share from the same pool of URL prefix names. Pick something memorable, as you will be using this value later. Use the **Check Availability** button to make sure the name you settle on is available for your use, then when you are ready, click on the blue **Next** button in the bottom-right of the view.



Check the box for **Create this environment inside a VPC**, as this Lab generated a VPC for your use, and we need to learn the extra step associated with configuring Elastic Beanstalk for use in a VPC. Do not try to **Create an RDS DB Instance with this environment**, simply click on the blue **Next** button in the bottom-right of the view.

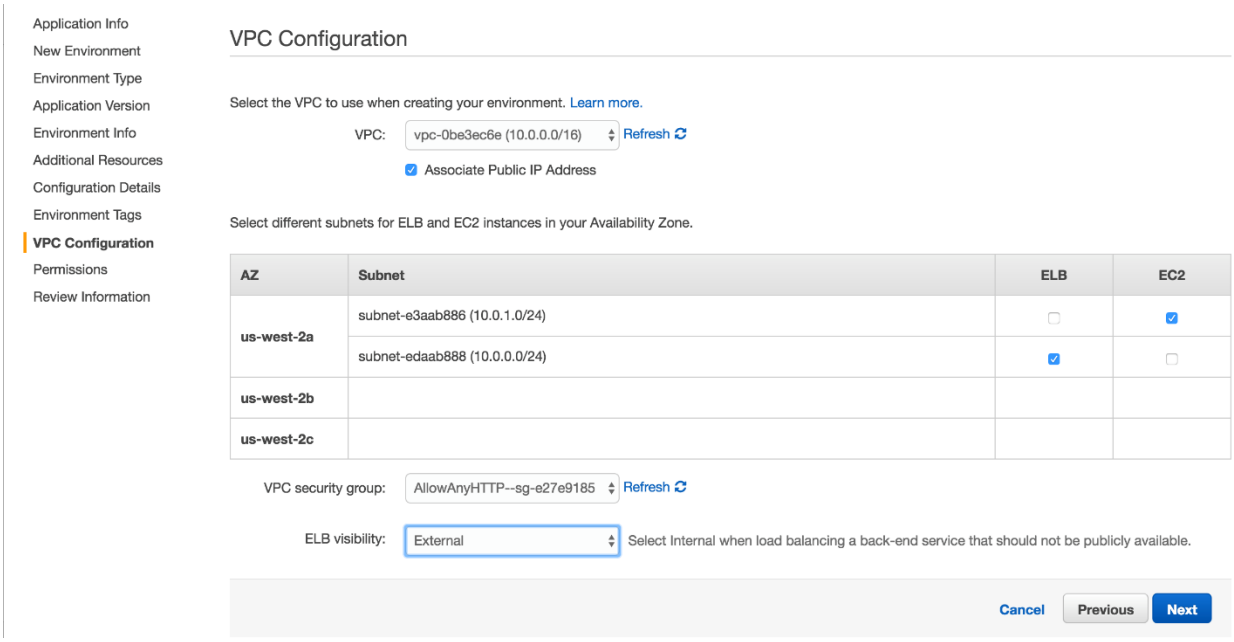


You will land on a large configuration page. The only value you should set for this lab is to changing the **Application Health Check URL** value from an empty text field to a single slash, to represent the root of the Elastic Beanstalk domain's structure (`/`). After making sure to set this value to `/`, scroll down to the bottom and hit the blue **Next** button in the bottom right of the view.

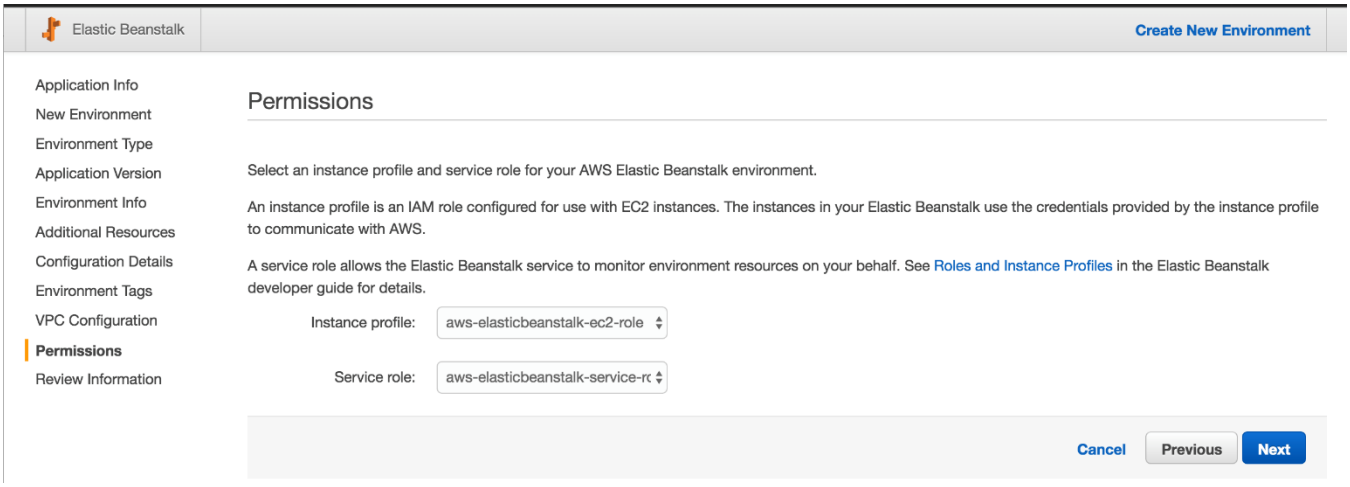


In the Environment Tags section click on the **Next** button, you should now have landed on the **VPC Configuration** view. Here you must select the proper subnet values for the EC2 and ELB portions of the Elastic Beanstalk deployment. The Lab will have auto-generated subnets with the same exact same CIDR blocks, so you should put the ELB in the Subnet with a CIDR block covering the lower range of the two, as this is the public subnet's block. When you're done evaluating your work, please proceed to the next view by clicking the blue **Next** button in the bottom right.

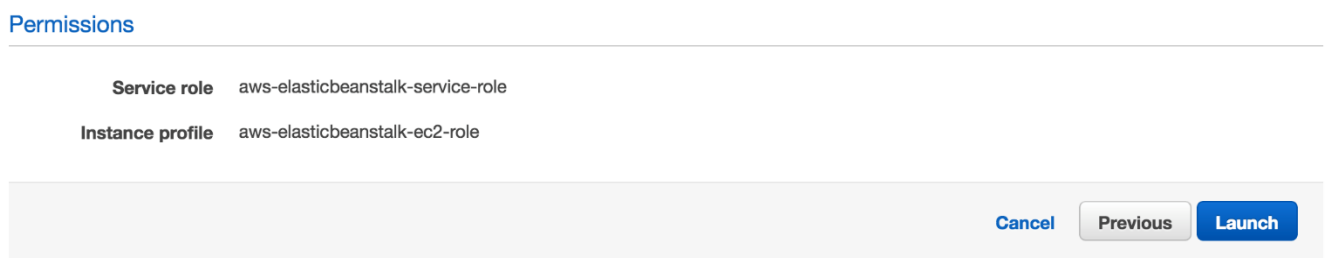
Note: You might have to select a different VPC in order to match the one showed in the laboratory.



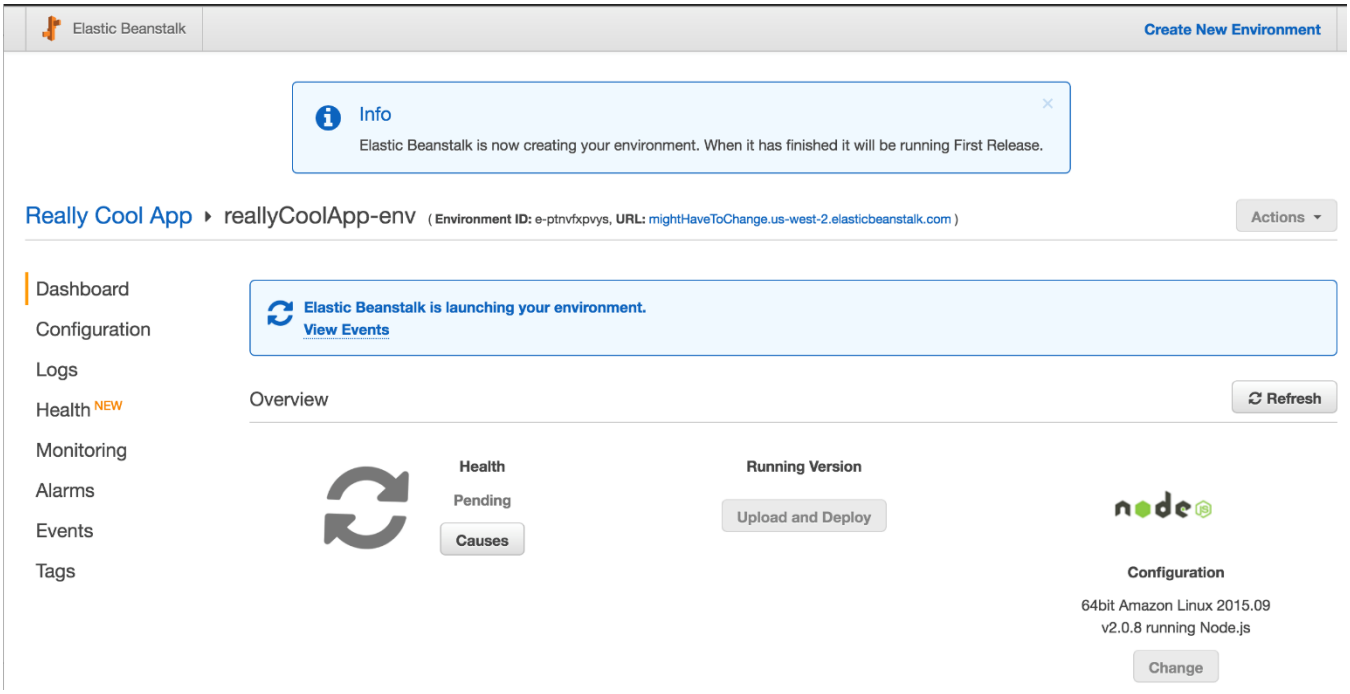
The next screen covers Roles and Identity on AWS. Use the standard suggested Elastic Beanstalk roles for the both the individual instances and the service roll.



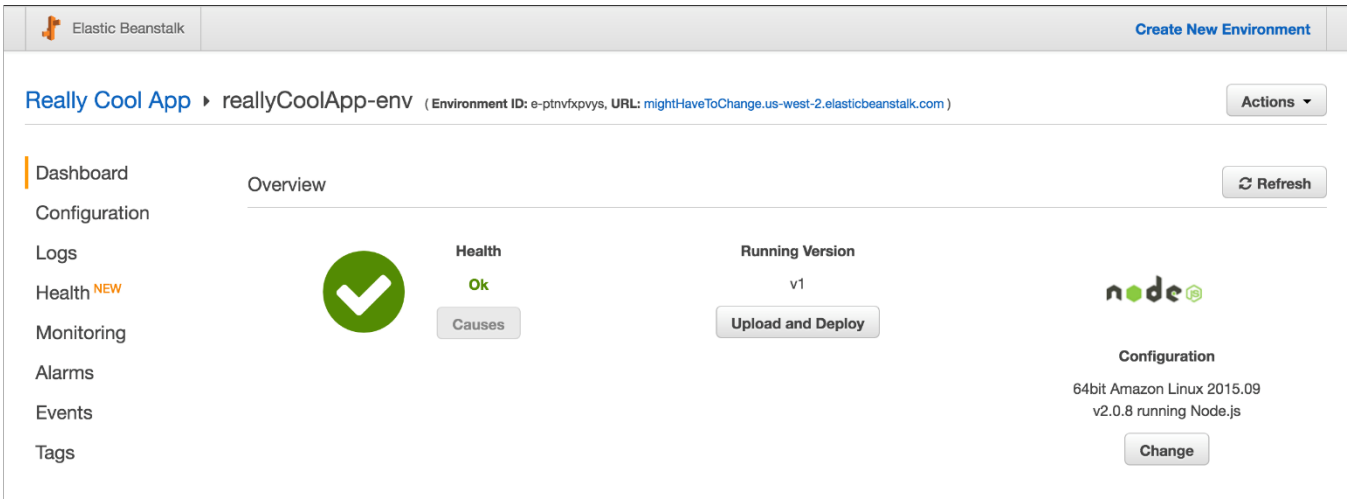
Scroll all the way down to the bottom of the view, then hit the blue **Launch** button.



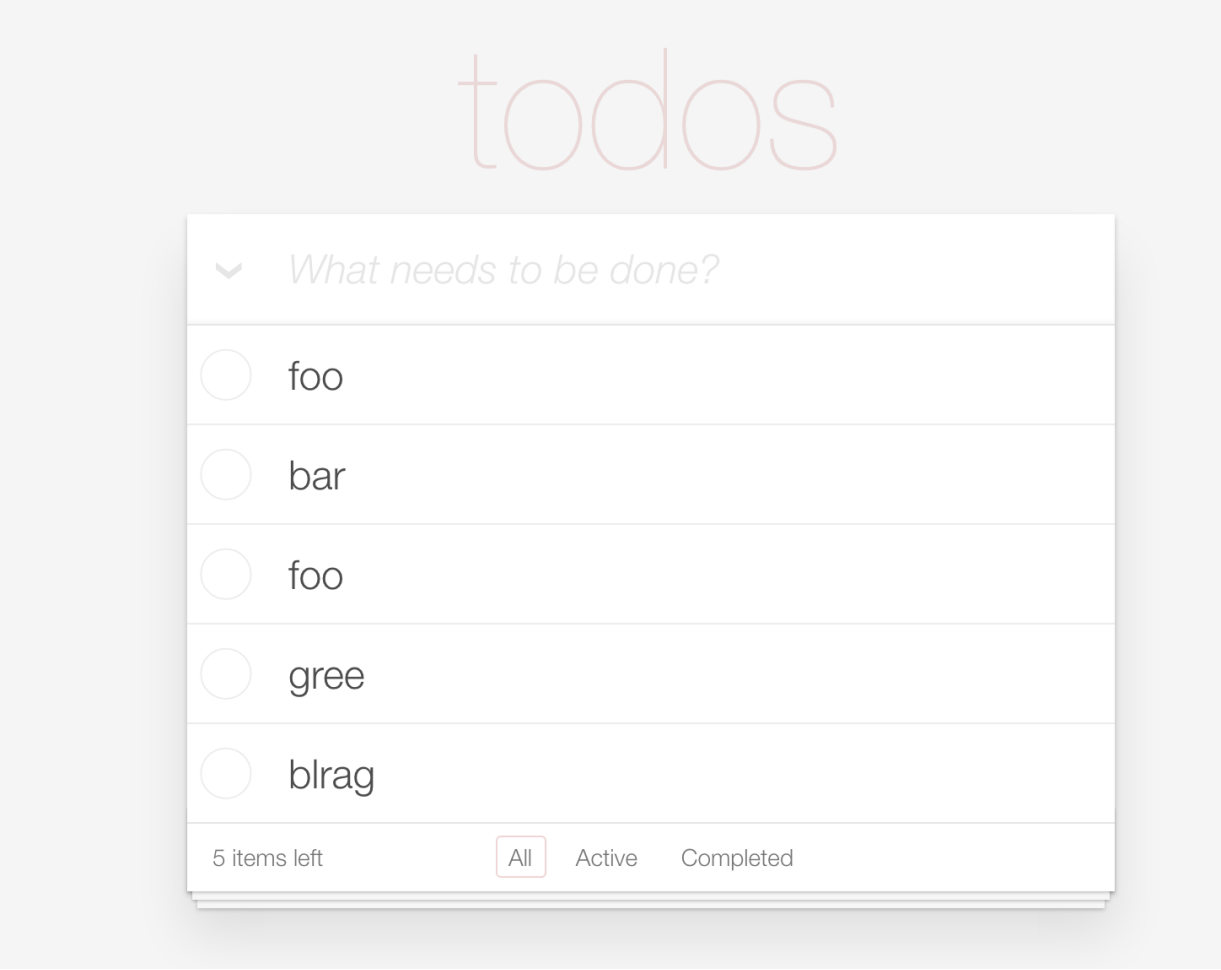
After hitting **Launch**, you should see something like the image below, substituting your own names for a couple of the fields...



... After waiting for a while, your system will transition into code green, like below.



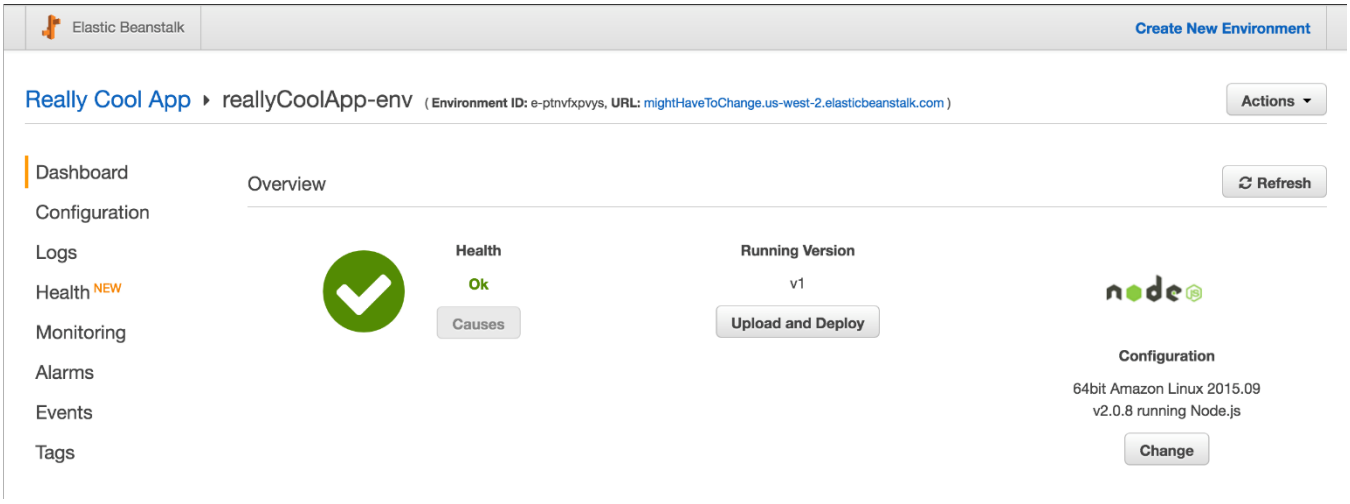
Once your system has signaled it's done launching, you should be able to enjoy the app itself. Go check it out!



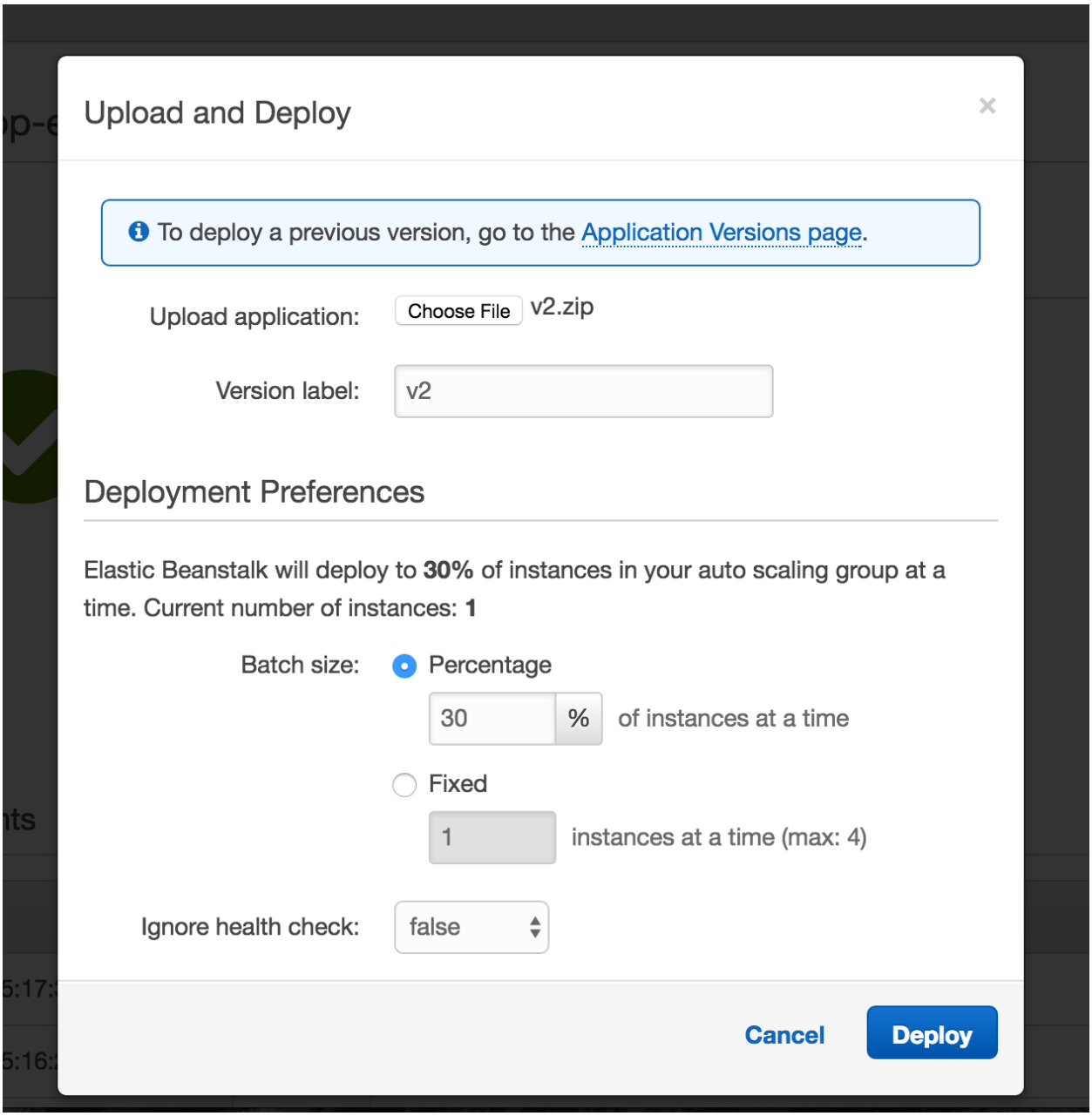
The application is a simple todo app. This Lab uses the Angular.js Todomvc.com mini app.

Step 2. Run a Rolling Deploy

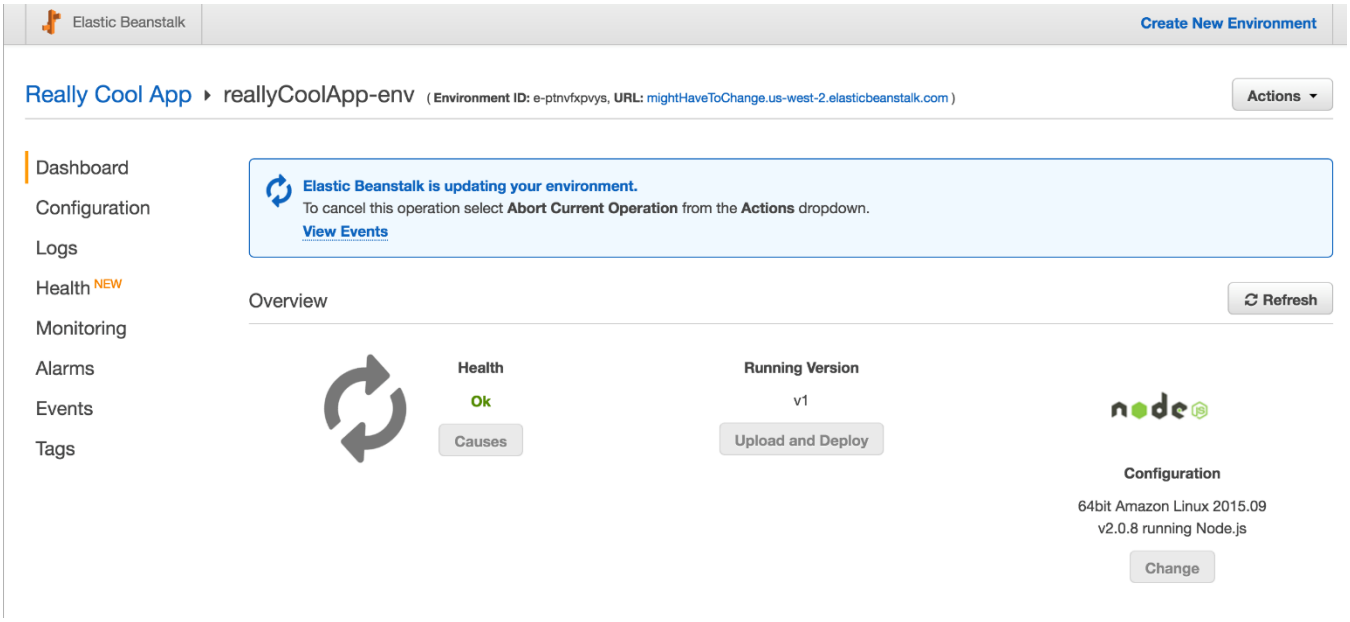
Once you have had a bit of time to play with the new environment we created, return to the status screen.



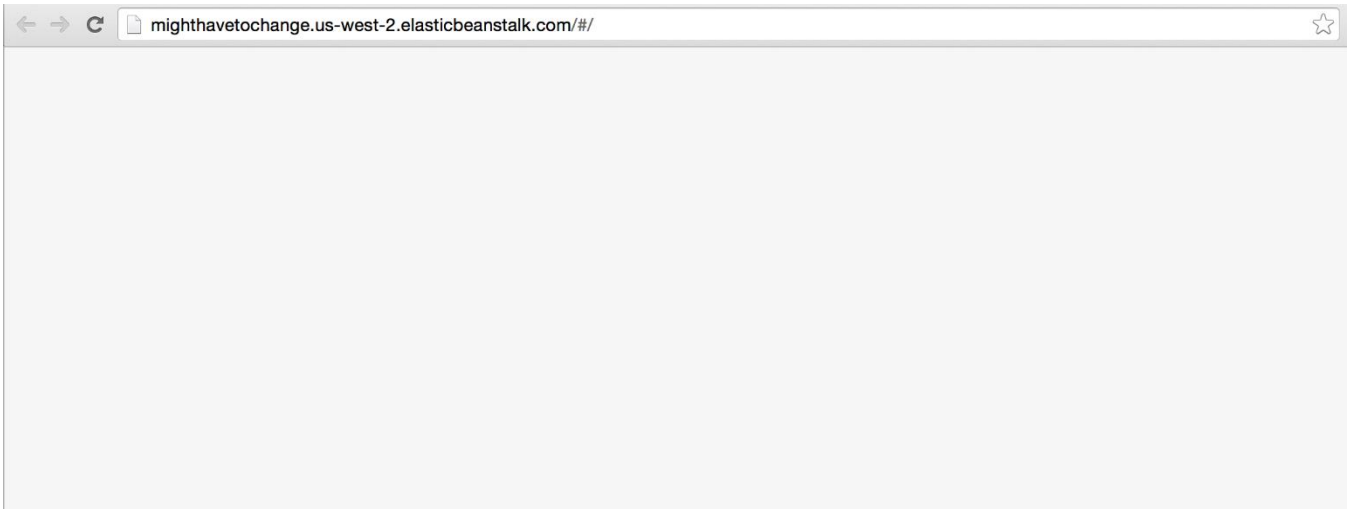
When the dialog opens, select this version two code and click on the **Upload and Deploy** button in the bottom-right corner of the screen.



After hitting deploy, Elastic Beanstalk will begin working on a Mutable Rolling Deploy of your code bundle. The environment will turn Gray, and will have a number of messages which amount to "please wait".

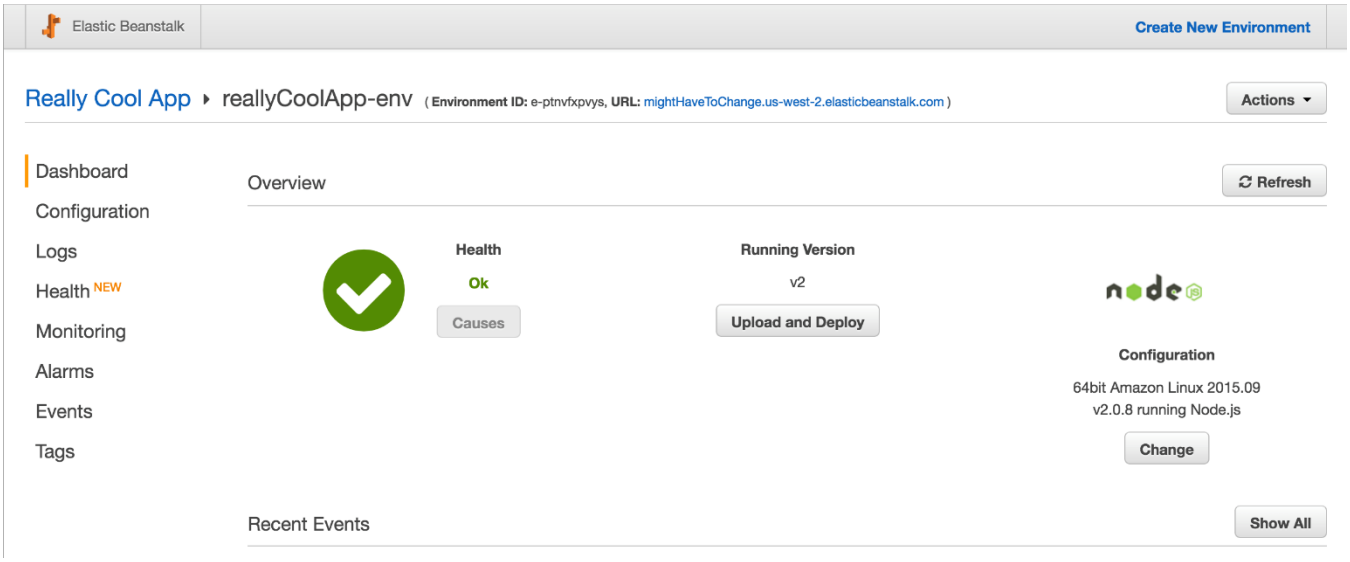


While you are waiting, think about how a rolling deploy works. If you watched the associated course for Advanced Deployment Techniques on AWS, you should know that a Mutable Rolling Deploy will update the environment by briefly shutting down portions of it during deploy. If you go back to the "todo" front-end, you may be able to refresh during this Rolling Deploy and find that the application is unavailable.



While the Rolling Deploy in Elastic Beanstalk is very simple and the most convenient one, it may not fit your use case.

After a while, your environment should stabilize. The gray spinner on the left of center will stop spinning and turn back into a green check mark.



Your environment should report that it is running V2 now. The only change we made in V2 was altering some text in the todo view from **todos** to **Cool V2**. Are you able to see the updated test when you re-visit the application environment URL?

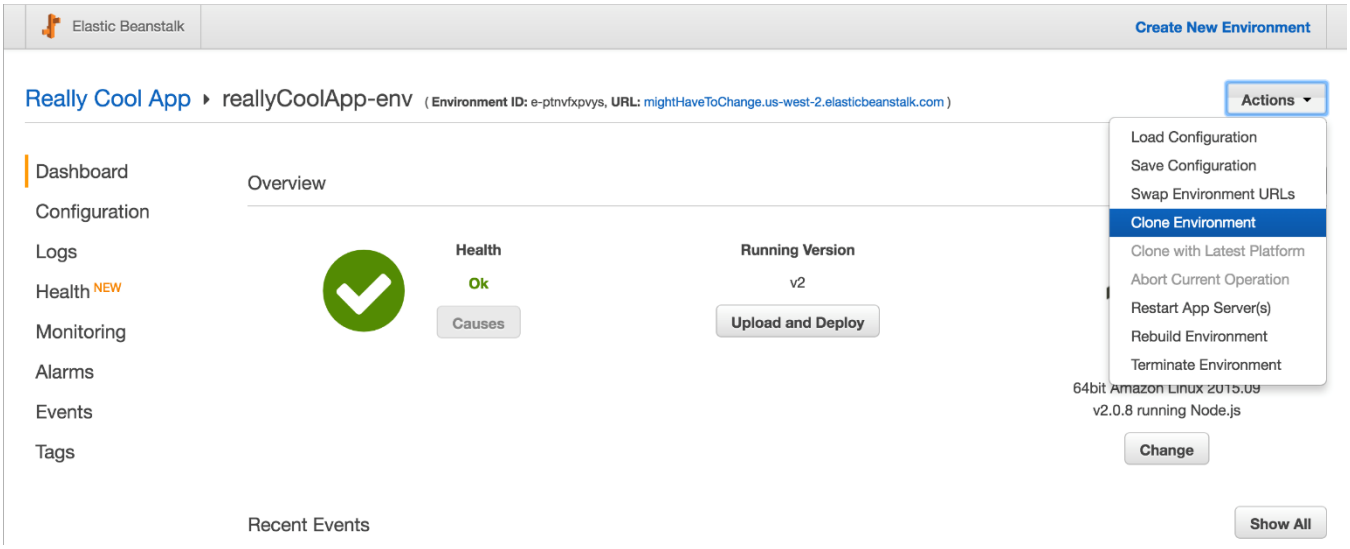
Once you confirm that you can see the changes, proceed to the **next step** in this lab.

Step 3. Prepare a Blue-Green Deploy

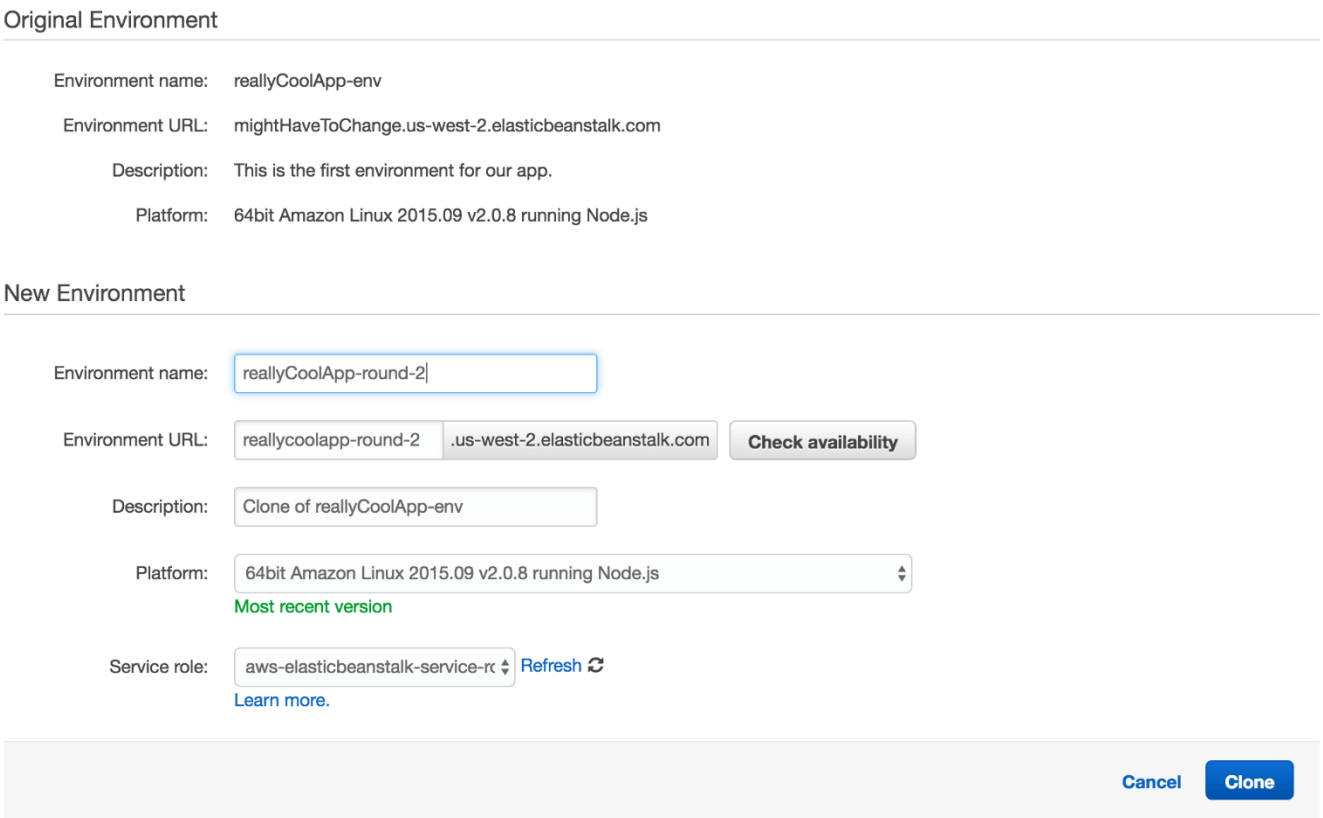
Now that we have run a Rolling Deploy, it's time to try our hand at the other major kind of deploy, the Blue-Green deploy.

If you recall, the first thing we need with this kind of deploy is a complete second environment that mirrors the one currently being used.

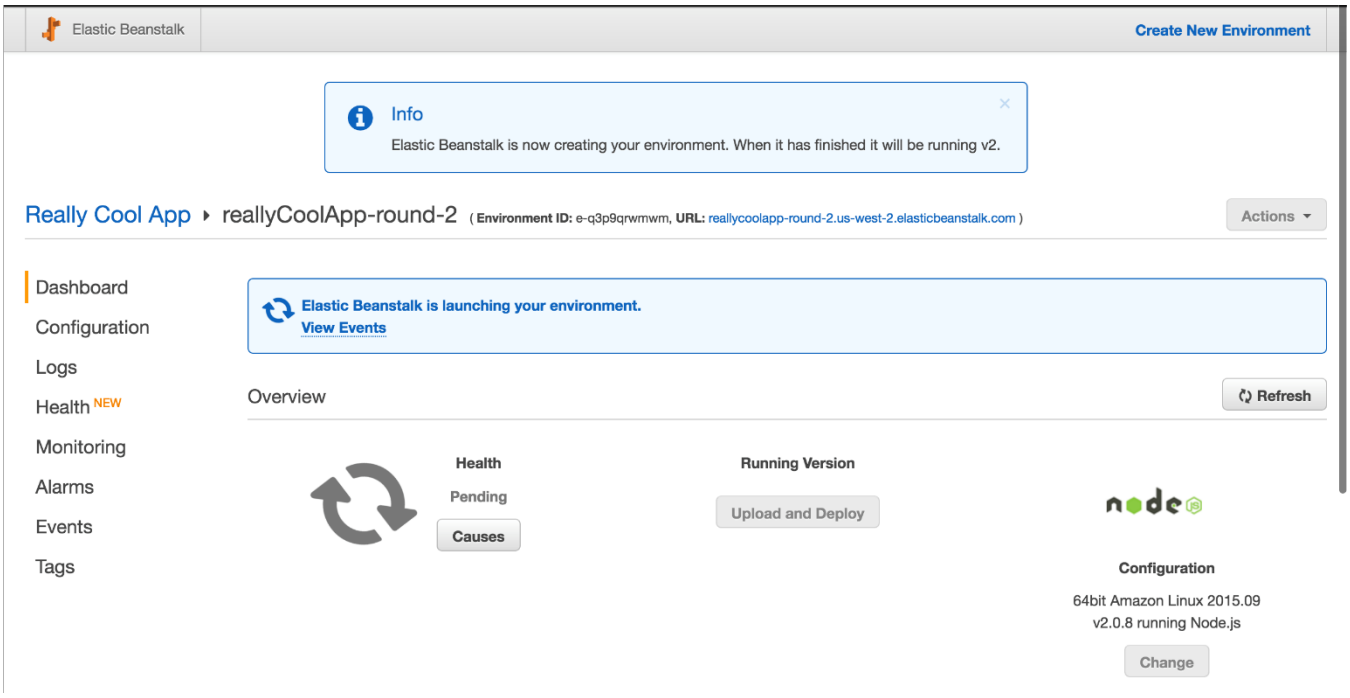
Fortunately, Elastic Beanstalk has a **Clone Environment** feature. From within the environment overview area, click on the **Actions** dropdown in the top right, then click on **Clone Environment**.

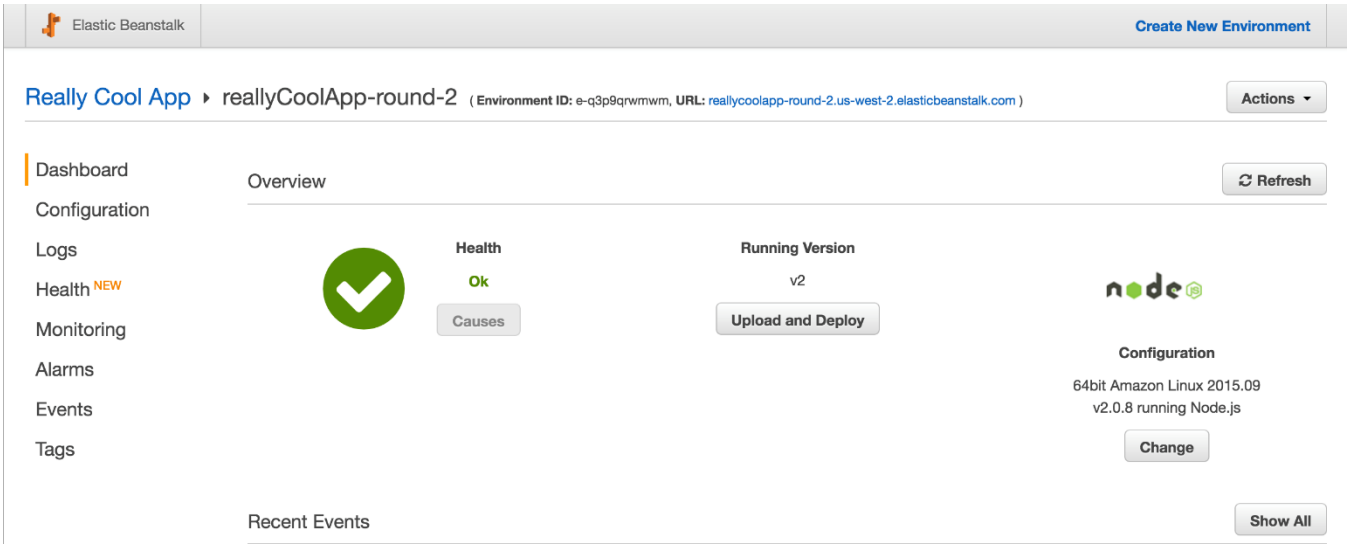


You will be presented with a form and a view that describes the existing environment at a high level, then also shows the settings of the to-be-created environment. We are going to use this as the new environment to try blue-green on, so pick a name and Environment URL which is self-explanatory.

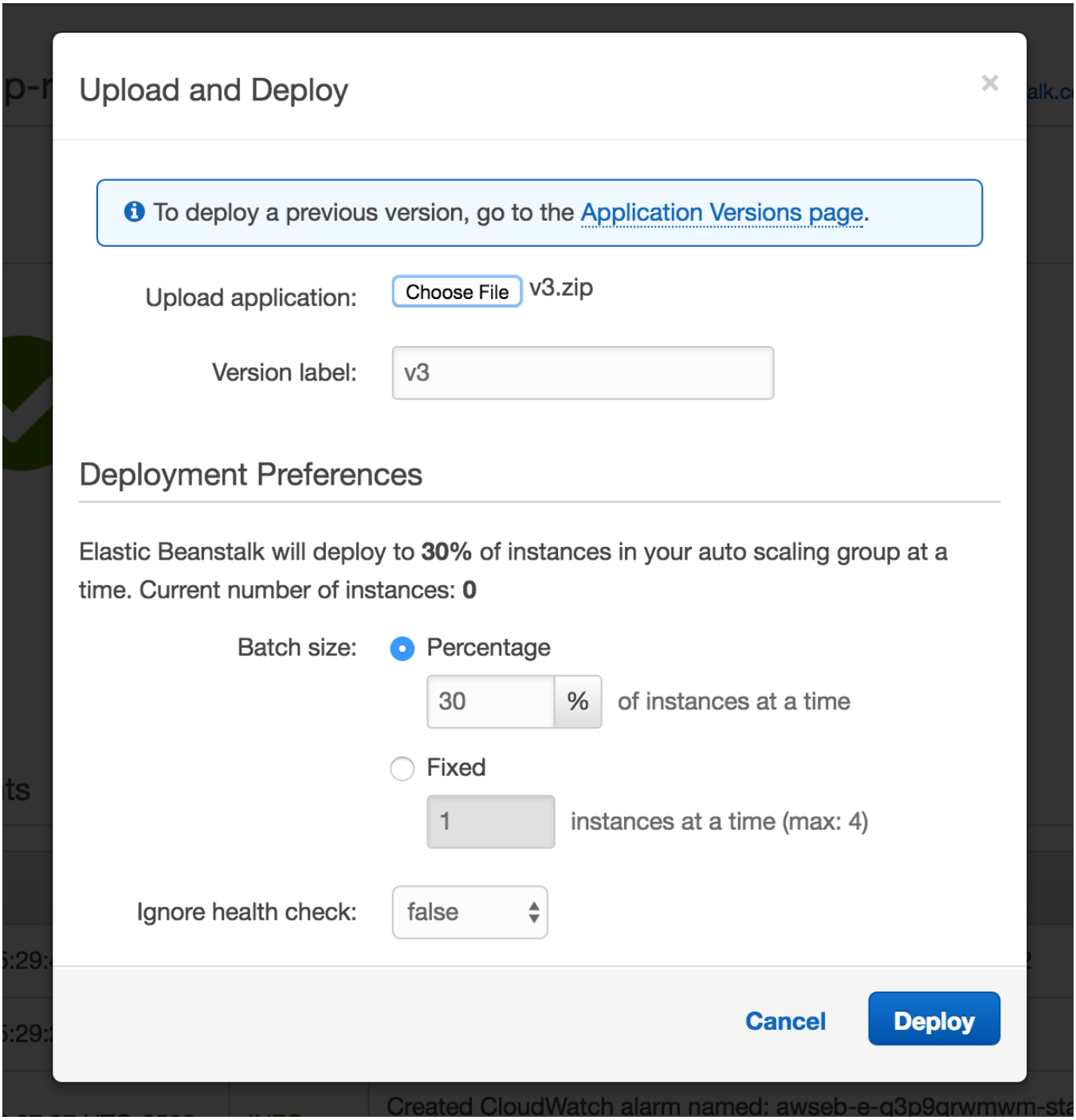


Once you click the blue **Clone** button, the same process and view that we experienced earlier during environment creation appears. Wait it out again.

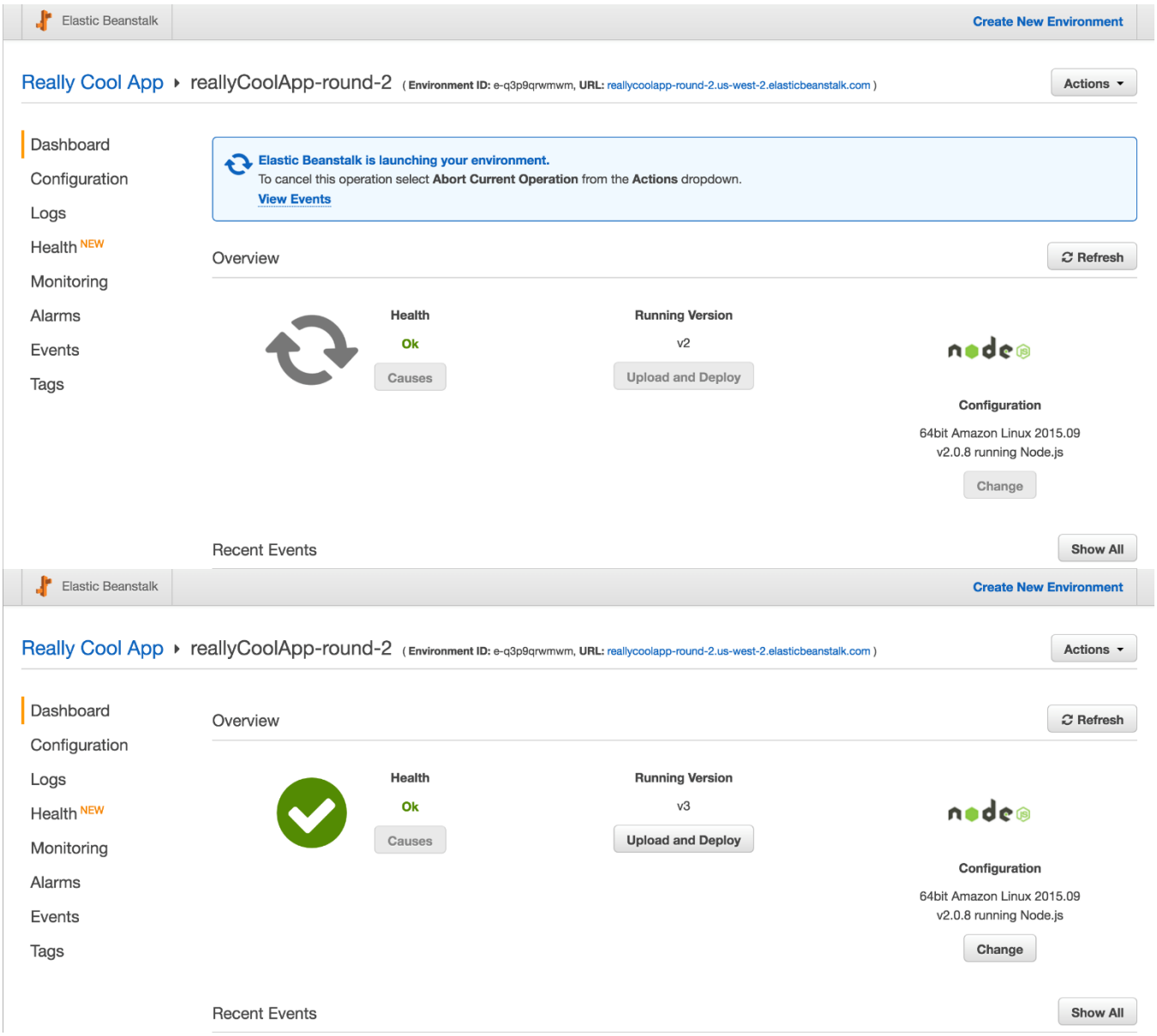




Once you have the second environment open, we need to prepare that environment to be the upgraded version, or version number 3. Open the code deployment dialog again, use this version three code, then execute a Rolling Deployment on this environment - because it is unused right now, we can tolerate the availability and performance concerns surrounding Rolling Deployments.



Wait again while this deploy completes.



Now that the application's third version is deployed, manually verify that the code was deployed correctly to this environment by navigating to the URL associated to the environment and view the functioning Todo application. Note that this time, we changed the header text again, to **B-G Deploy**.

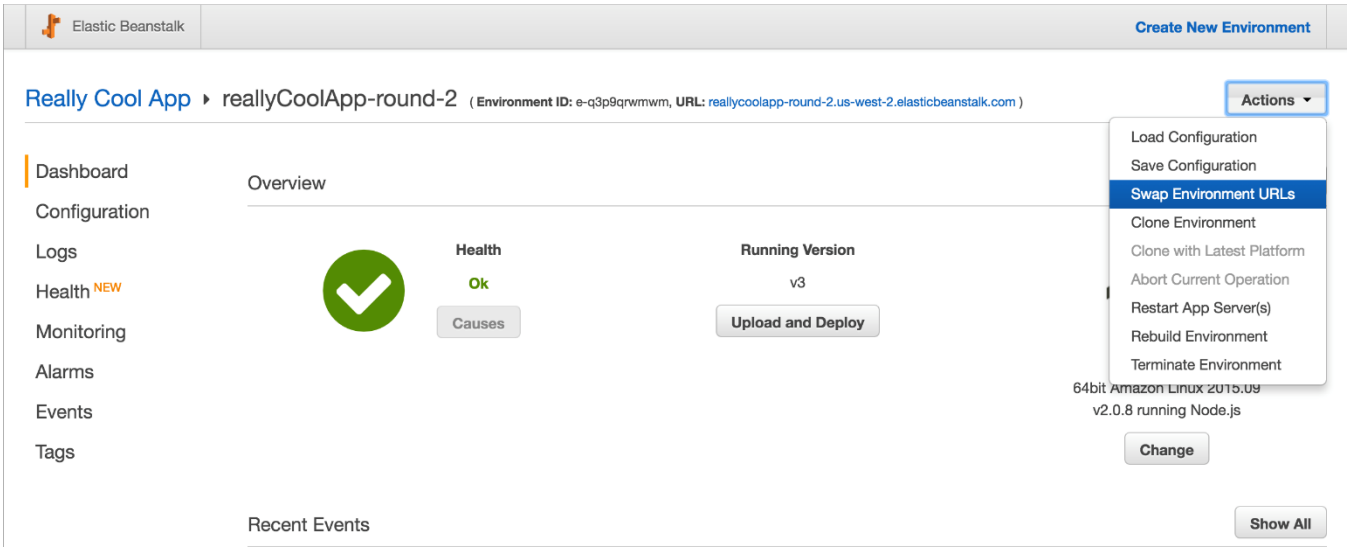
Once you're satisfied that a third version of the same application has been deployed, proceed to the **next step** in this Lab.

Step 4. Do the Deploy Swap

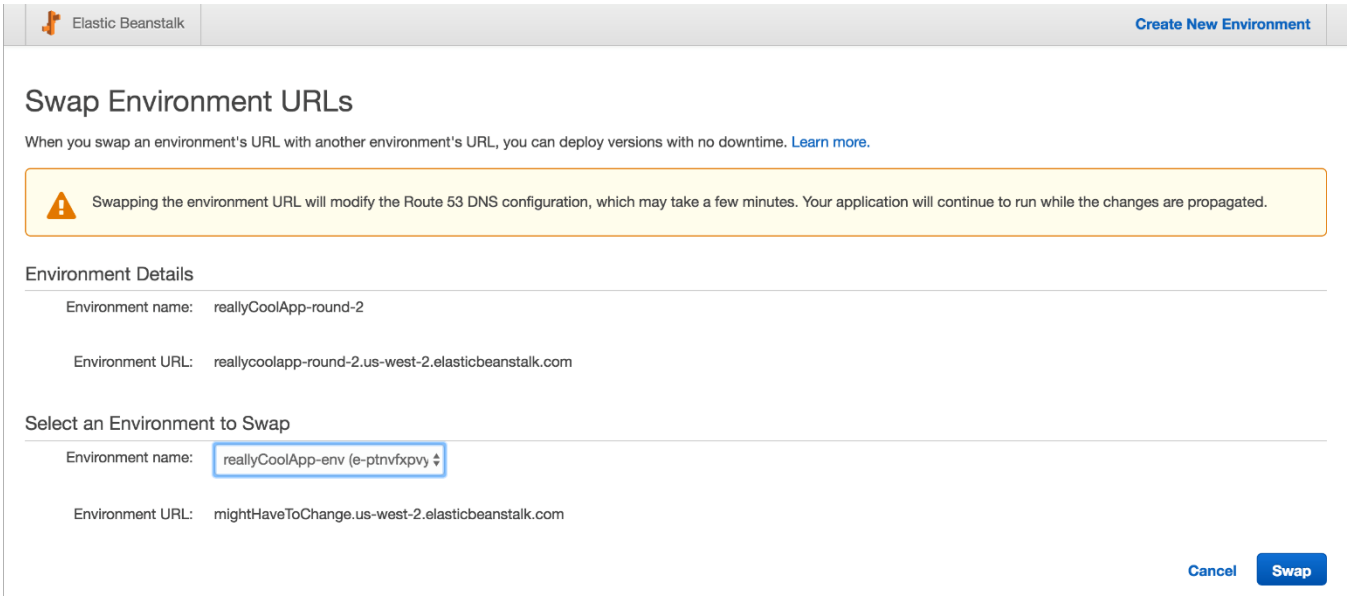
When you are ready to run the rest of the Blue-Green deploy, navigate back to the page with the environment dashboard.

Blue-Green deployments rely on having entire systems or subsystems deployed all at once using a DNS or reverse proxy cutover. We currently have two environments, one of which is running V3 and the other of which is running V2. The one running V2 is using the original DNS that we need to continue using, so we should quickly and invisibly trade DNS on the V2 and V3 versions, so V3 can get the original and more valuable URL.

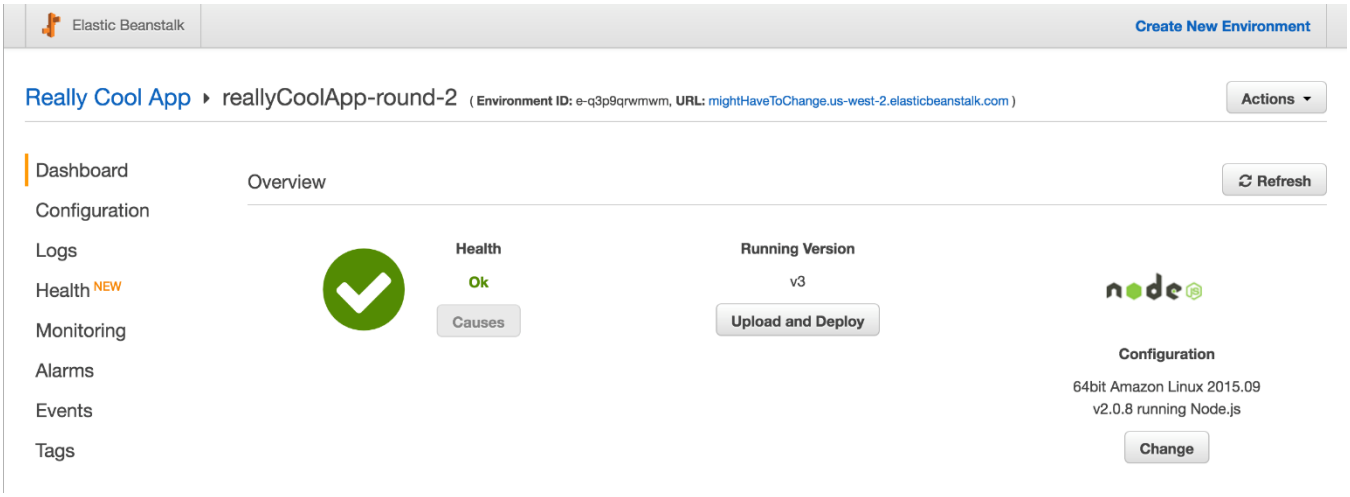
Click on the **Actions** dropdown, then click on **Swap Environment URLs**.



You will be taken to a page which allows you to select the environment you would like to swap with. Pick the V2 environment.

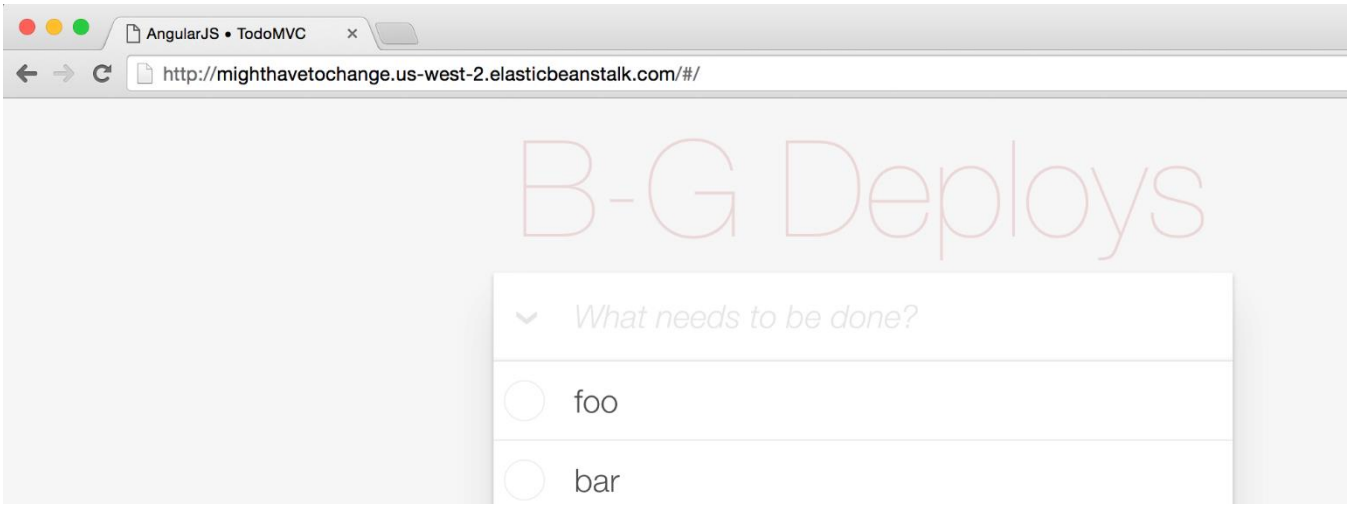


While the system warns you that the change could take a number of minutes, it's effectively instant in nearly all cases. The environment will likely be finished swapping as soon as you return to the dashboard for either environment.



This swap is simple to perform and fast. There is no downtime during the cutover since the DNS had both environments working correctly. All traffic to the original Elastic Beanstalk Environment is now routed to the new one running V3, as the V3 environment has taken over all the traffic via DNS.

Try it out - visit the original URL you made for the **first** environment, and you should see the V3 version running there now.

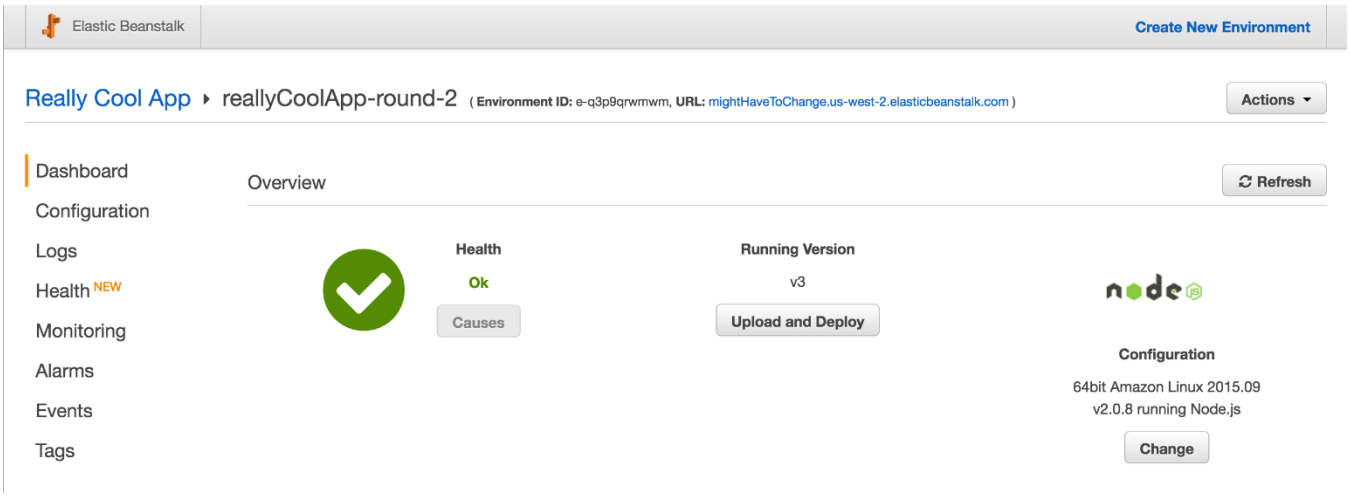


Once you are satisfied that the deployment worked as intended, proceed to the **next step**.

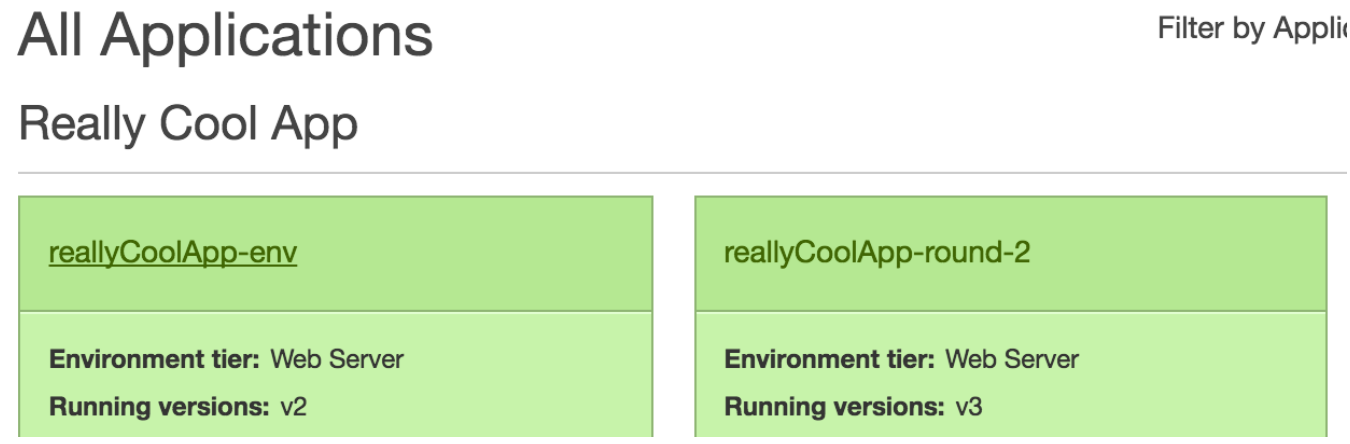
Step 5. Clean Up Old Resources

Now that we have successfully tried two kinds of low-downtime cloud deployment techniques on Elastic Beanstalk, we might be concerned about the costs associated with these techniques. The deploys are inexpensive, but they may get a little more expensive if we simply leave the environment we swapped away from online all the time. We should clean up the environment to save costs.

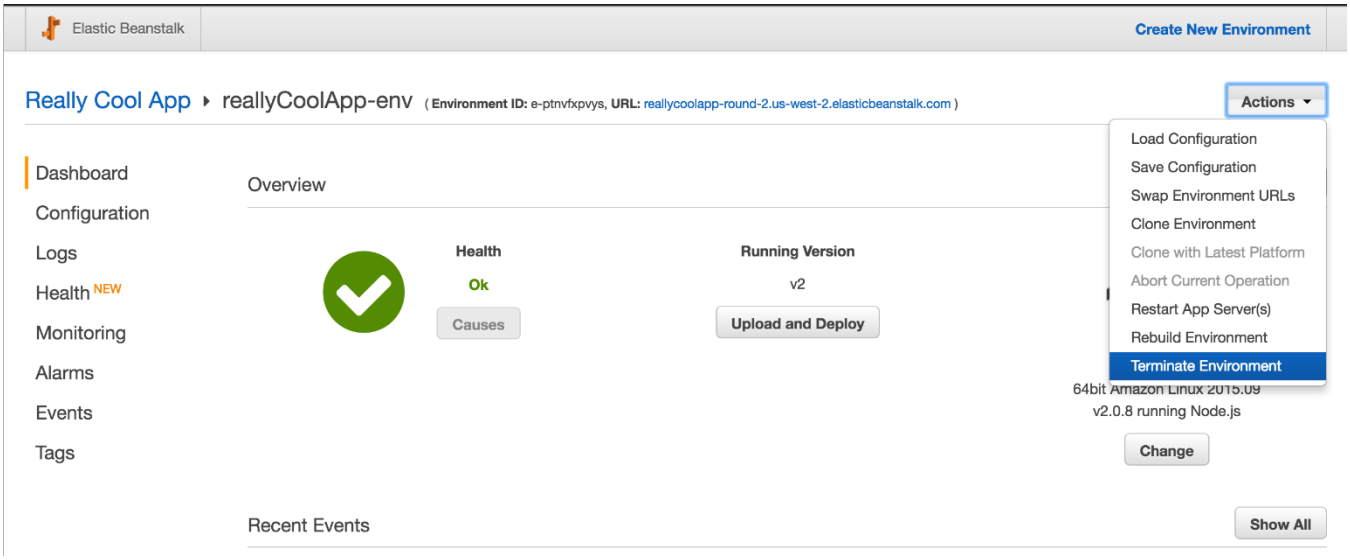
First, navigate back to the dash for the environment we just swapped to - the one running V3.



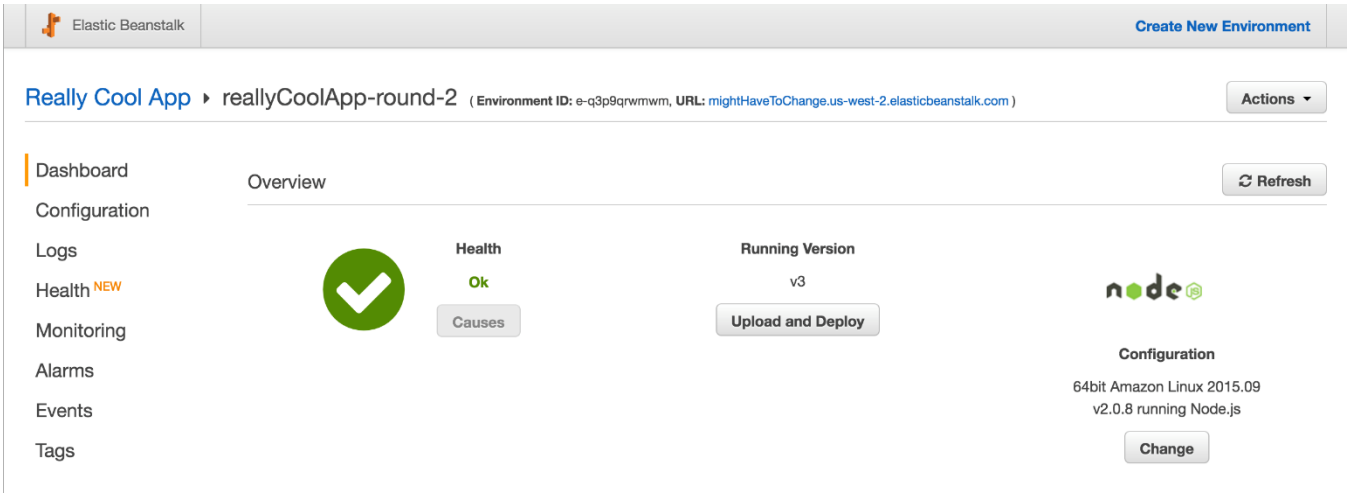
Click on the Elastic Beanstalk icon in the top-left of this view to navigate to the **All Applications** view. Your application will show up, along with two green boxes, representing two live environments associated with the application.

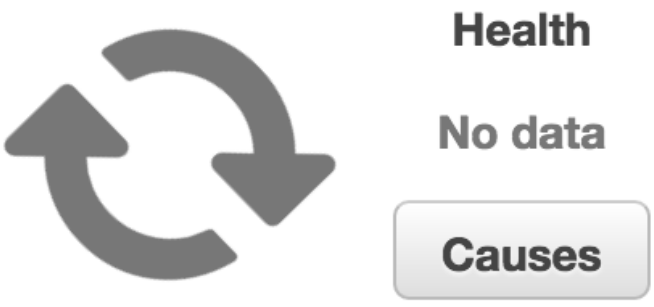


Click the one running **v2**. You will be taken to the dashboard for this now-obsolete environment. Click on the **Actions** drop-down, then click on the **Terminate Environment** option, so we can destroy this environment and save some money.



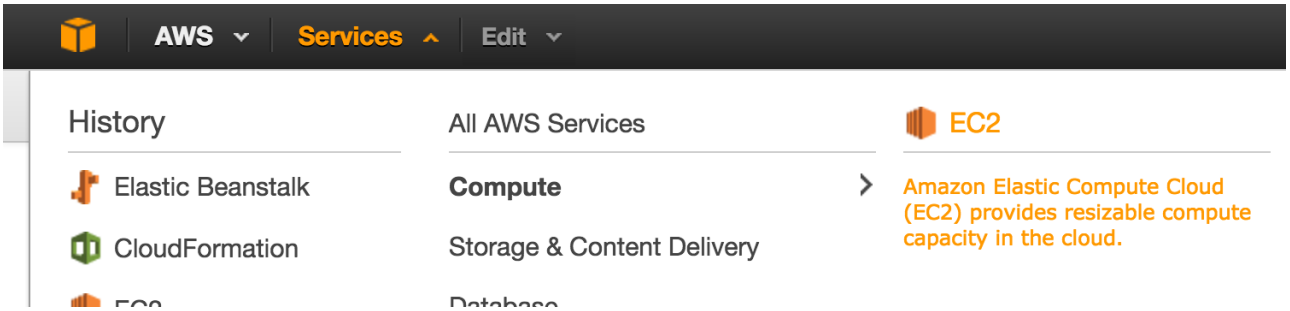
You should see the environment spin into a gray state, then see the **Health** spin into a death spiral - it might swing between **OK**, **Severe**, and **No Data**. Wild!



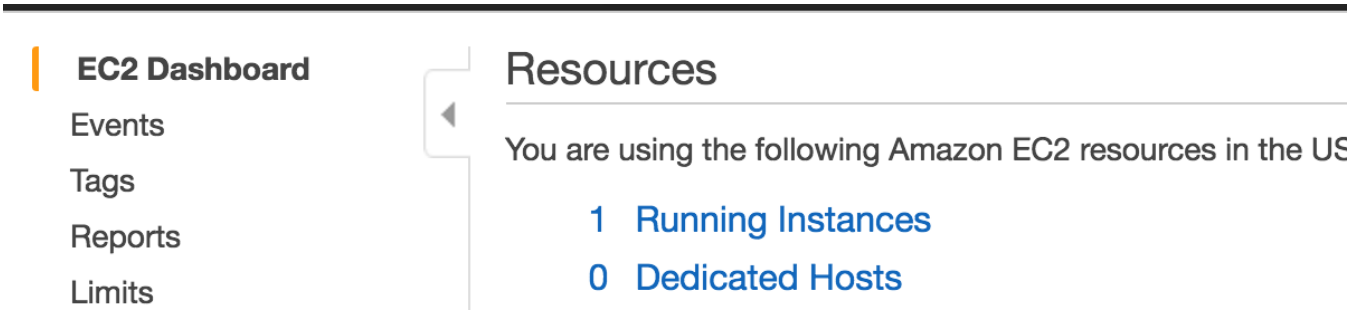


Eventually, this will end, and you will be sent back to the **All Applications** view. This time, there will not be two green environments under your application - there will be one light-gray one, and one green one. The light-gray one is the terminated environment.

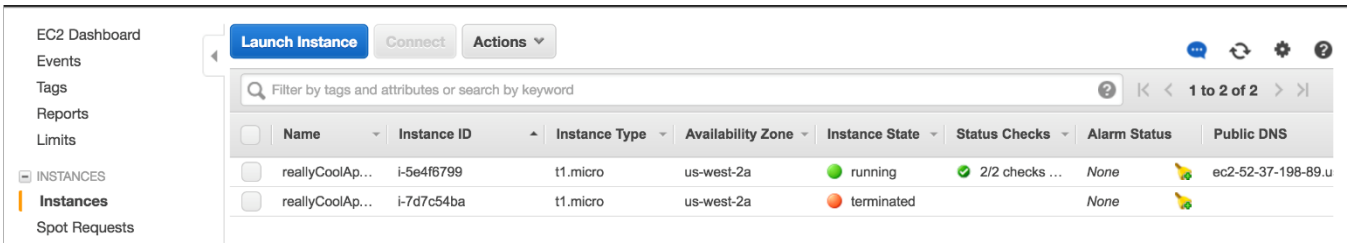
To verify that we reduced the spend of our system, we can enter the EC2 console and make sure that the EC2 instances that Elastic Beanstalk creates on your behalf are terminated. Navigate to the EC2 Console by clicking on the **Services** dropdown in the top-left of the Console, hovering over the **Compute** section, and clicking on the **EC2** section.



After navigating to the top level of the EC2 console, you should be able to see that there is only 1 Running Instance now if you allowed your old/second Elastic Beanstalk Environment to totally terminate.



For further verification that actual instances were terminated, click the **Running Instances** link and review the table of instances. One should be **Running**, and one or more should display as red and **Terminated**. This is / these are instance(s) which belonged to the now-terminated Elastic Beanstalk Environment.



You've learned how to run two types of low-interruption, high-automation deploys on AWS Elastic Beanstalk.