**Creating an Elastic Beanstalk App & Environment**

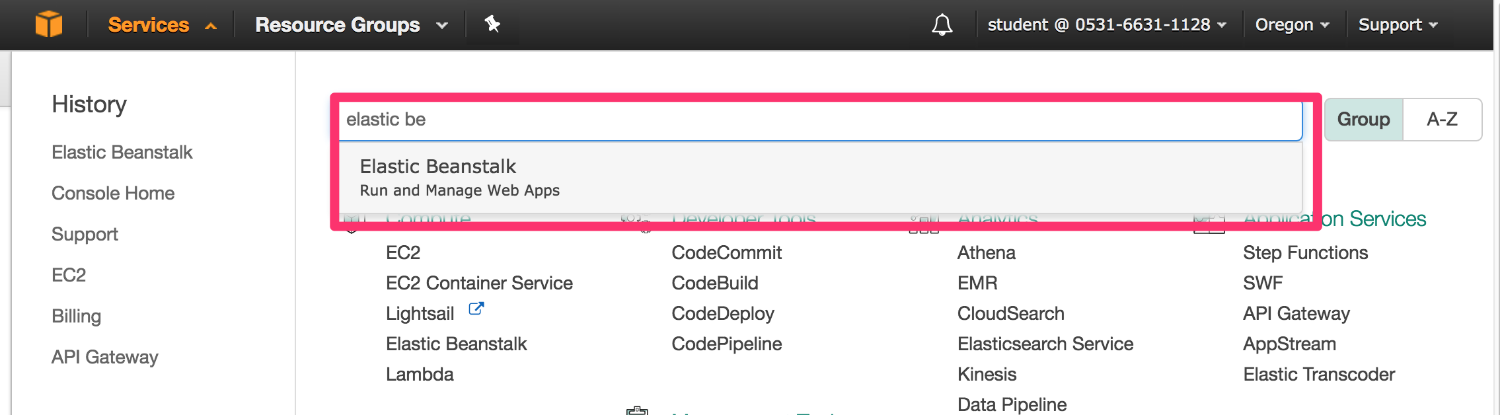
**Introduction**

Since you 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.

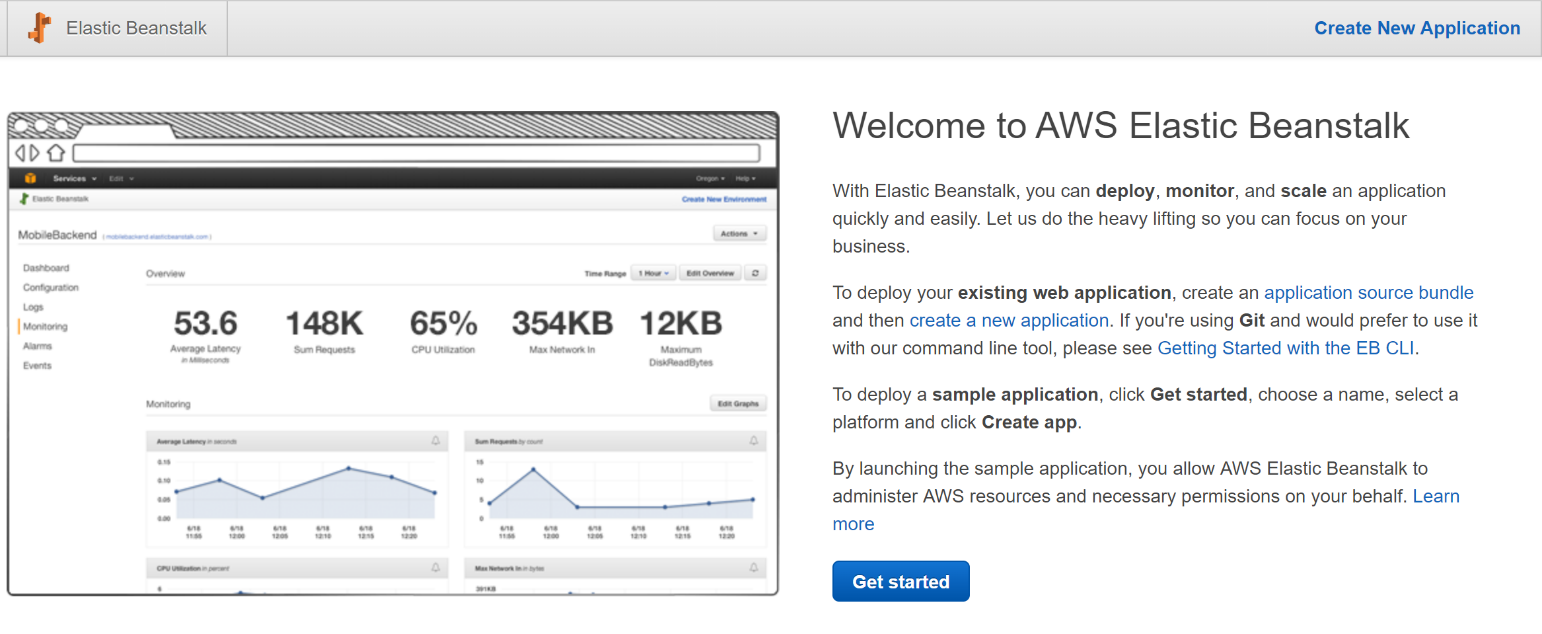
**Instructions**

1. [Click here to download the source code for the sample application](https://bitbucket.org/toorroot/devops_base/src/55d92e51a77912283c574969ab004461f54824cf/Labs/ElasticBeanstalk/?at=master) that you will use for this Lab.

2. Navigate to the AWS Elastic Beanstalk Console, by clicking on the **Services** in the top-left of the Console, entering *elastic beanstalk* in the search bar, and clicking on **Elastic Beanstalk**:



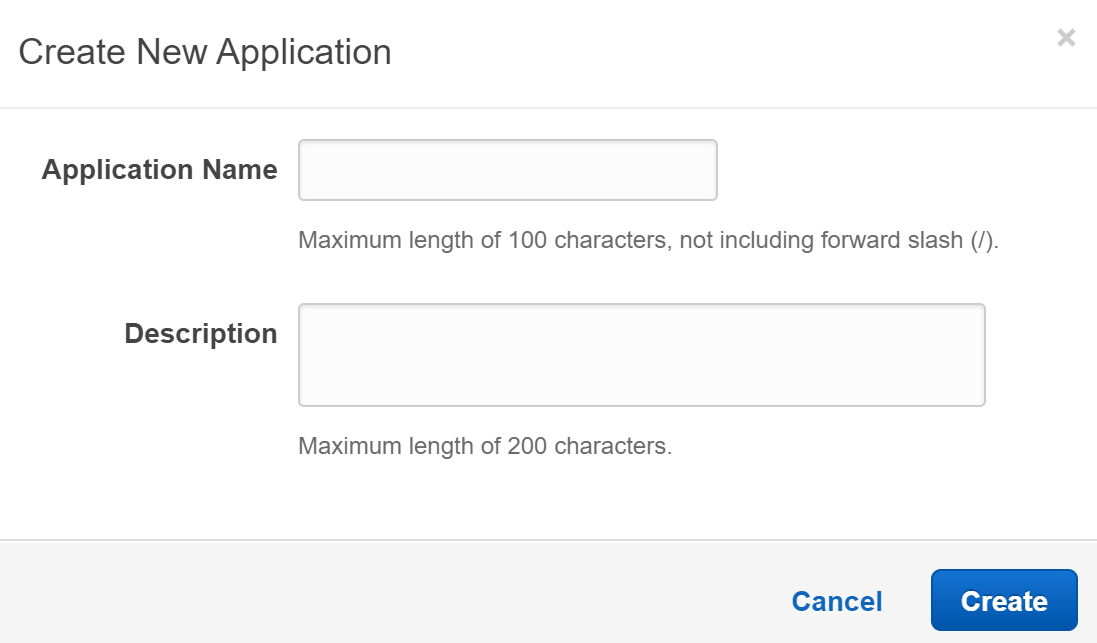
Because the account the Lab uses has never used Elastic Beanstalk before, you are presented with a welcome message:



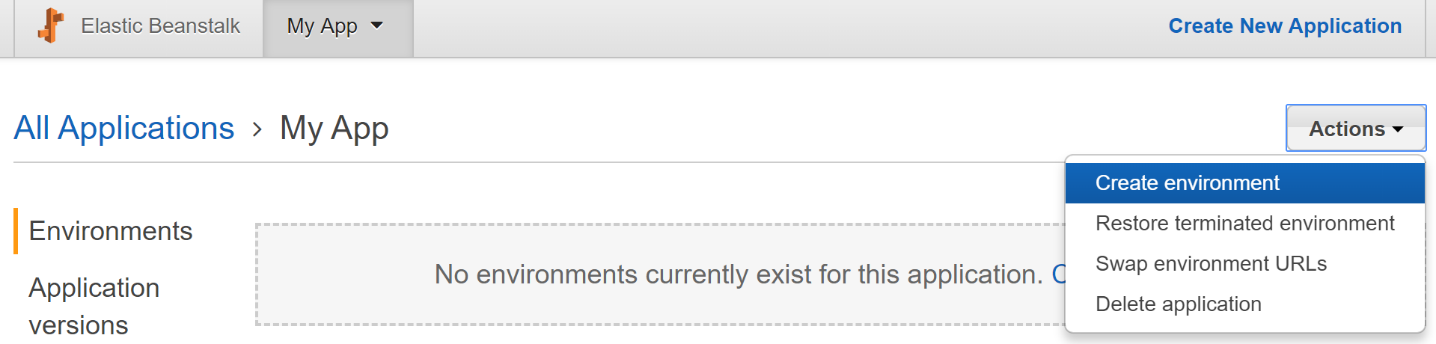
3. Click on the **Create New Application** link in the top-right of the Console view.

4. In the **Create New Application** pop-up, enter the following values and click **Create**:

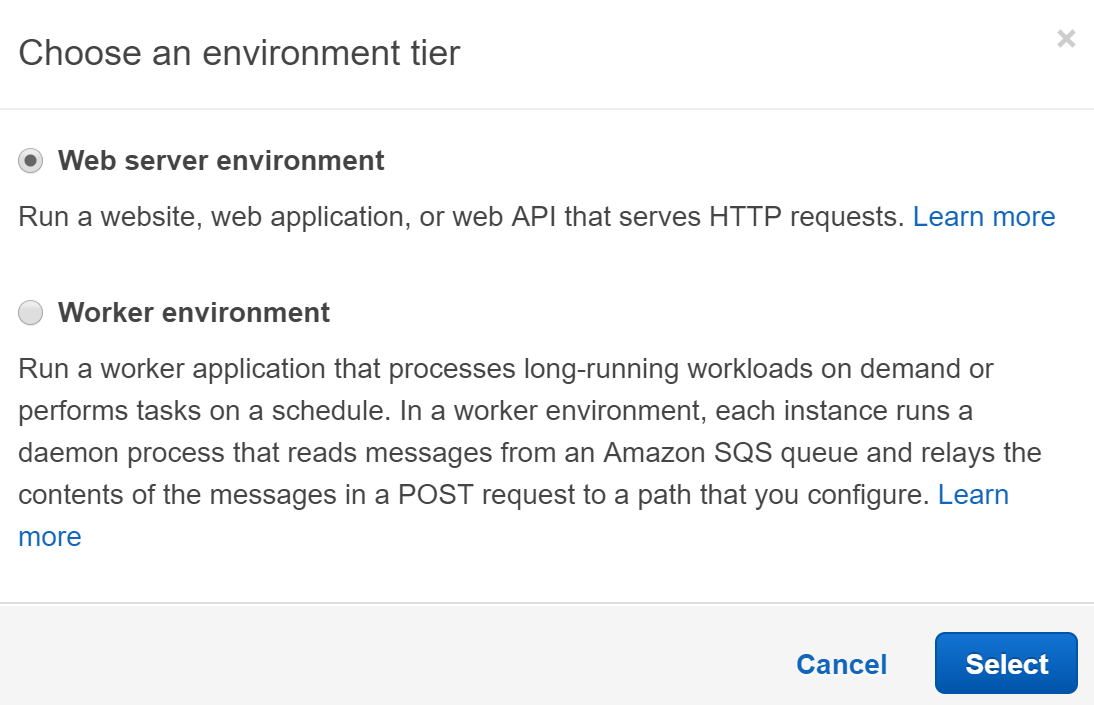
* **Application Name**: *My App* (The name doesn't affect anything and is only used as a label for you to distinguish between applications)
* **Description**: *Made in my Devops# account*



5. From the upper-right **Actions**menu, select **Create environment**:



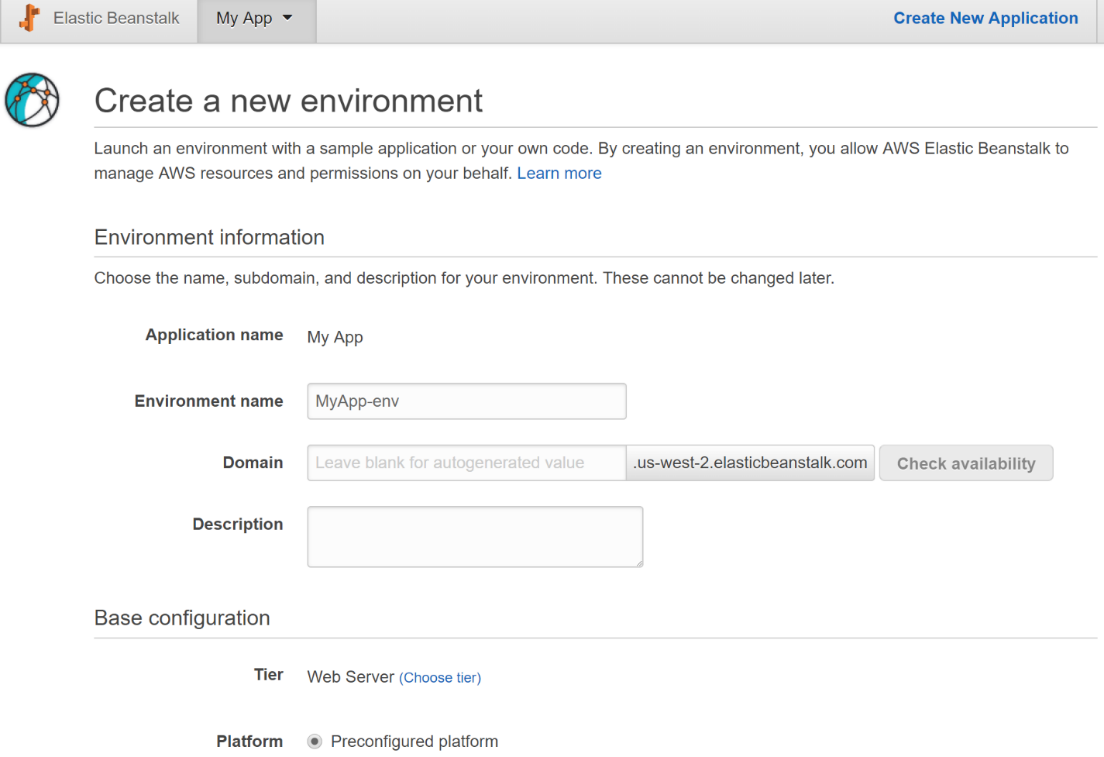
 6. **Select** the **Web server environment**in the **Choose an environment tier**form

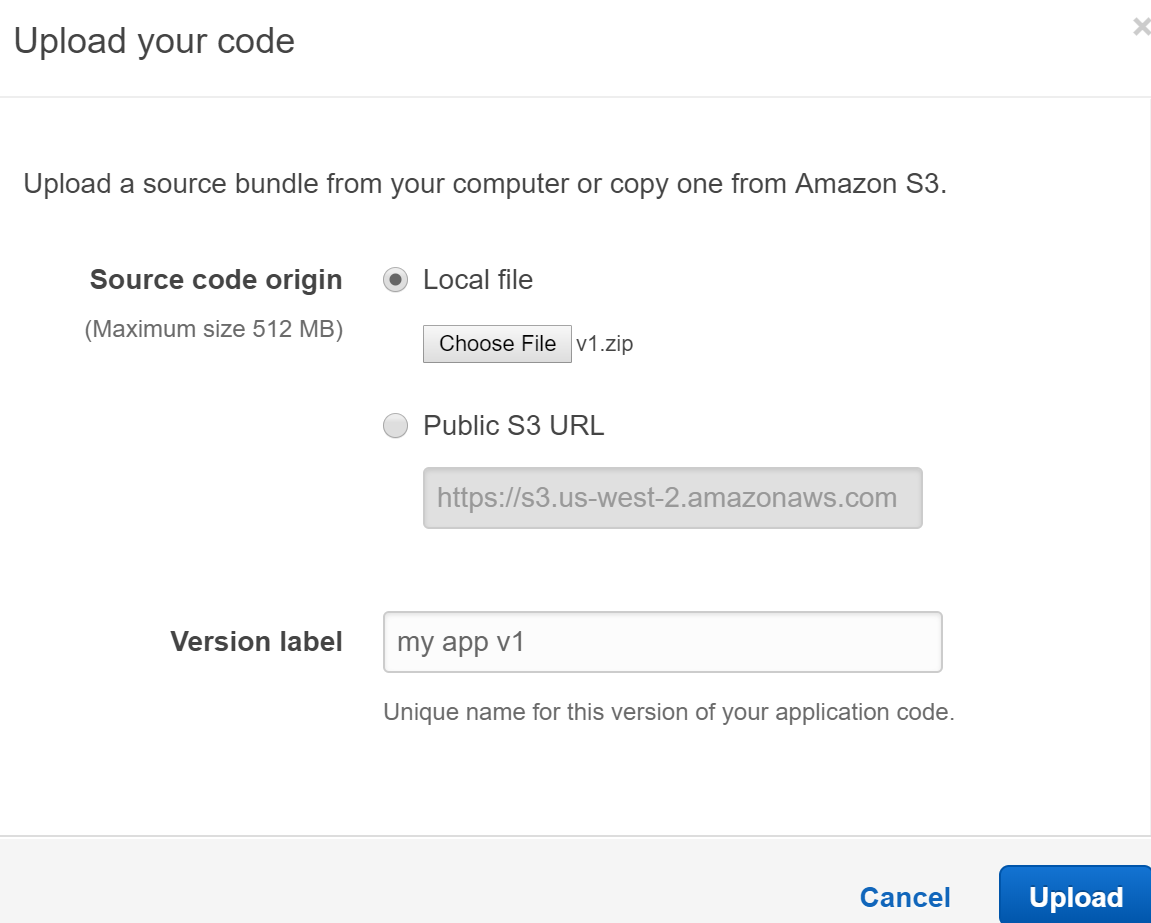


You will create a web application that responds to HTTP requests, so the **Web server environment** is the correct choice. The **Worker** **environment** is suitable for long-running or scheduled work and does not directly interface with browsers.

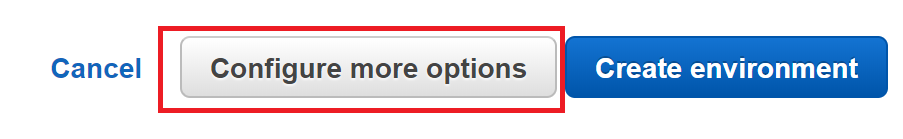
 7. In the **Create a new environment** form, enter the following values, accepting the defaults when not specified:

* **Base Configuration**
  + **Preconfigured platform**: Node.js (The code this Lab provides you is all Node.js/JavaScript)
  + **Application** **code**: Sample application (You will upload modified PHP code after first deploying the sample application)
  + **Upload your code**: Click **Upload** to open the **Upload your code** panel
    - **Source code origin**: Local file, then click **Choose file** and select the source code zip file you downloaded earlier
    - **Version label**: *my app v1*
    - Click **Upload**

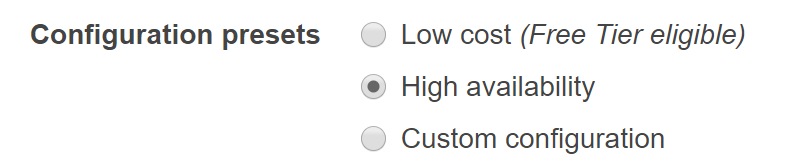




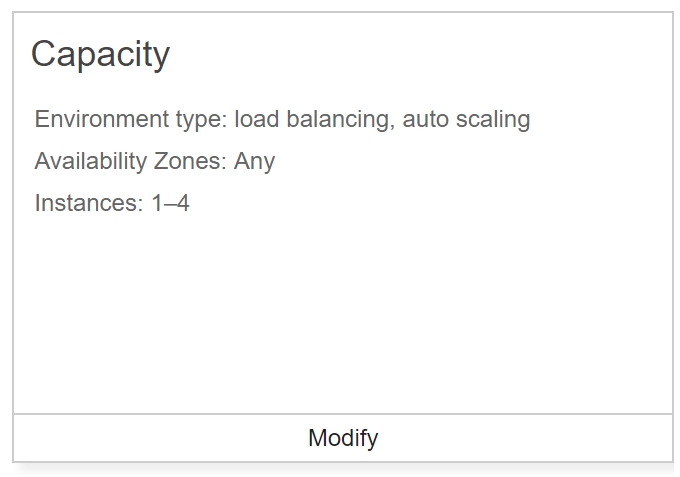
8. Click **Configure more options**:



9. Change the **Configuration presets** to **High availability**:

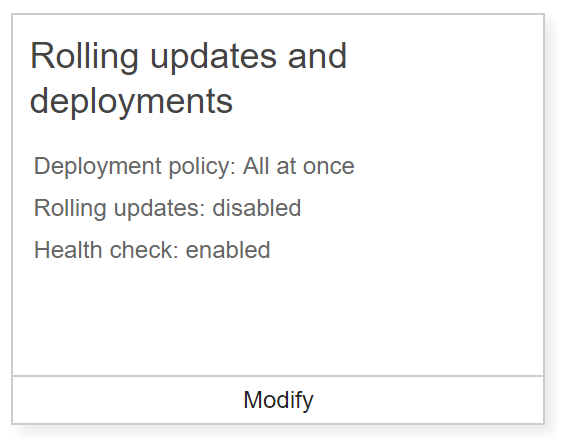


The **High availability** preset configures **load balancing** and**auto scaling**, as can be seen in the **Capacity**configuration card:



This is the correct configuration for this Lab and usually what you want in production environments. You will need to do some configuration adjustments to some of the other categories.

 10. Click **Modify** on the **Rolling updates and deployments** card:



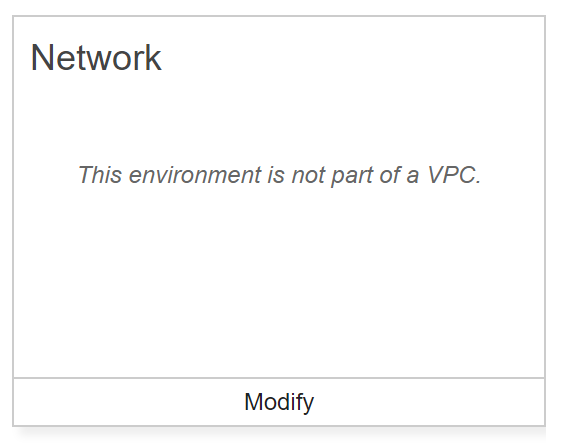
The default configuration deploys updates to the instances **All at once**. This causes downtime for the application which is not acceptable for production environments.

 11. On the **Modify rolling updates and deployments** form, enter the following values and click **Save** (accept default values when not specified):

* **Application deployments**
  + **Deployment policy**: Rolling
  + **Batch size**: 3*0* % (This configures how many instances an update is rolled out to at a time. 30% would roll the update out to the entire fleet in four rounds having around 30% reduction in available capacity)

The **Configuration updates** section configures how Elastic BeanStalk will handle deploying environment changes such as changing EC2 instance size. Such changes aren't included in the Lab so you can leave the default settings.

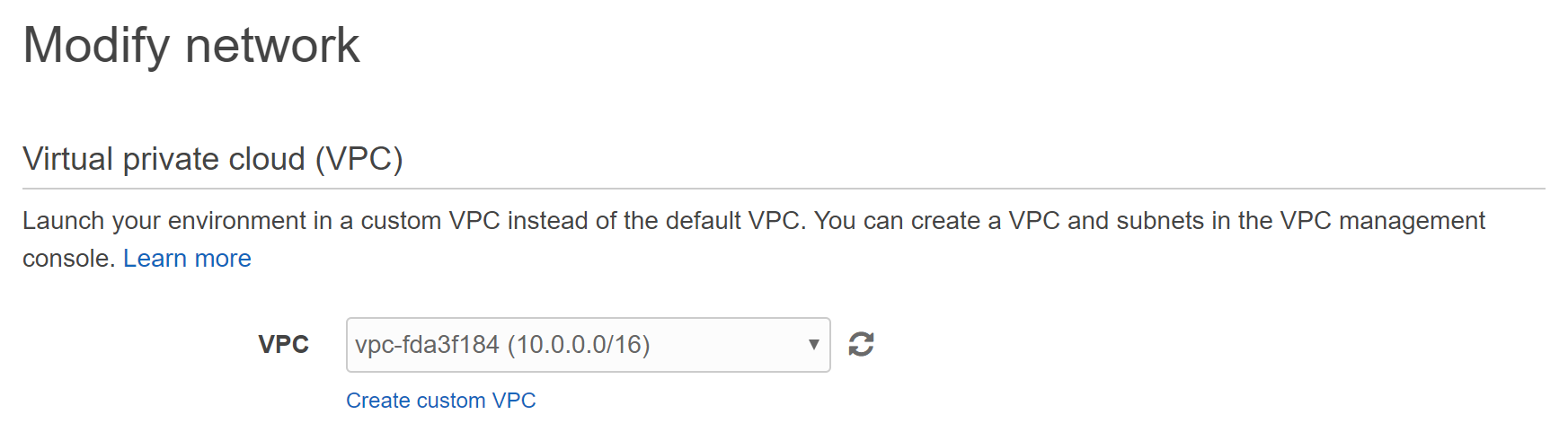
 12. **Modify**the **Network** configuration:



Having a separate VPC for the application makes it easier to manage.

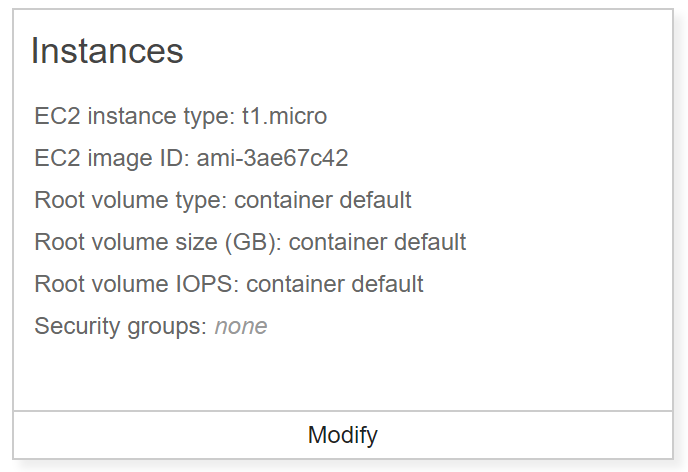
13. On the **Modify network** form, enter the following values and click **Save** (accept the default values when not specified):

* **VPC**: Select the VPC with a CIDR block of 10.0.0.0/16 (not the default VPC)
* **Load balancer settings**
  + **Load balancer subnets**: Check the subnet with **CIDR** block **10.0.0.0/24** (This is a public subnet)
* **Instance settings**
  + **Instance subnets**: Check the subnet with **CIDR** block **10.0.1.0/24** (This is a private subnet)



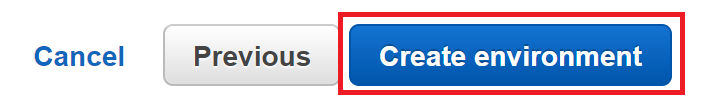
The network settings make the load balancer publicly accessible and do not assign public IP addresses to the EC2 instances running the application code. This makes the application only accessible through the load balancer from the internet. Because there is only one availability zone **us-east-2a**, the application could not tolerate an availability zone outage. You would want to use multiple availability zones in practice but for the sake of this Lab, one is acceptable.

 14. Observe the **Instances**configuration card:

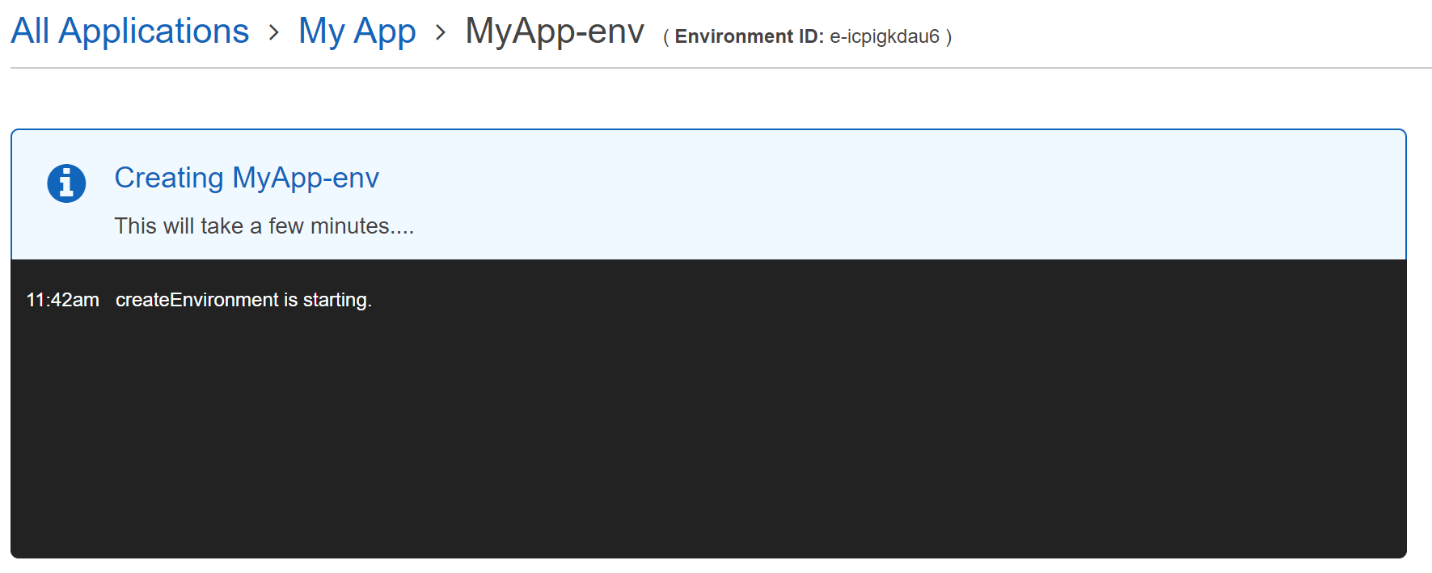


The **Security groups** has a value of ***none***, which may lead you to think the instances will not allow any inbound traffic. When you don't choose an existing security group, Elastic Beanstalk will create new security groups automatically for both the instances and the load balancer allowing inbound traffic on port 80. Because of this, it is not a problem to leave the security group set to ***none***.

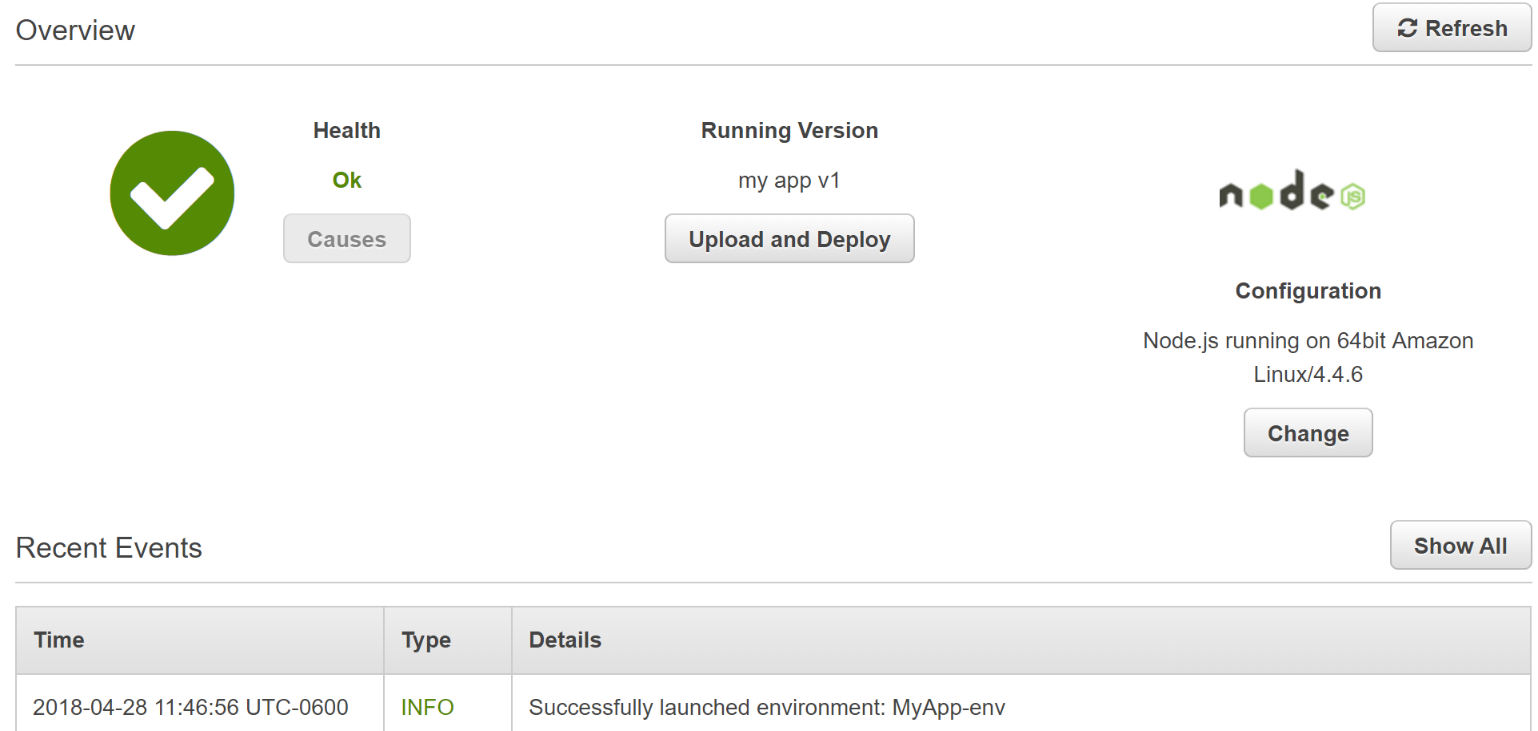
15. Accept all the other default settings, and click **Create Environment**:



You are taken to the environment's page where you can watch the progress of the environment creation process:

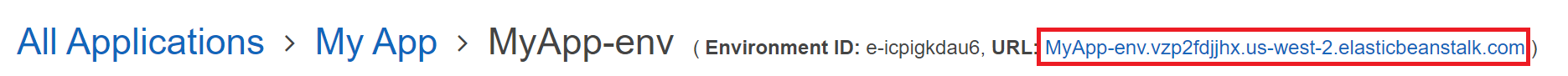


Wait for the environment to finish creating before proceeding on to the next Lab Step. You can understand a lot about how Elastic Beanstalk works by reading the messages. It takes about four minutes and you will see the following view when it is complete:

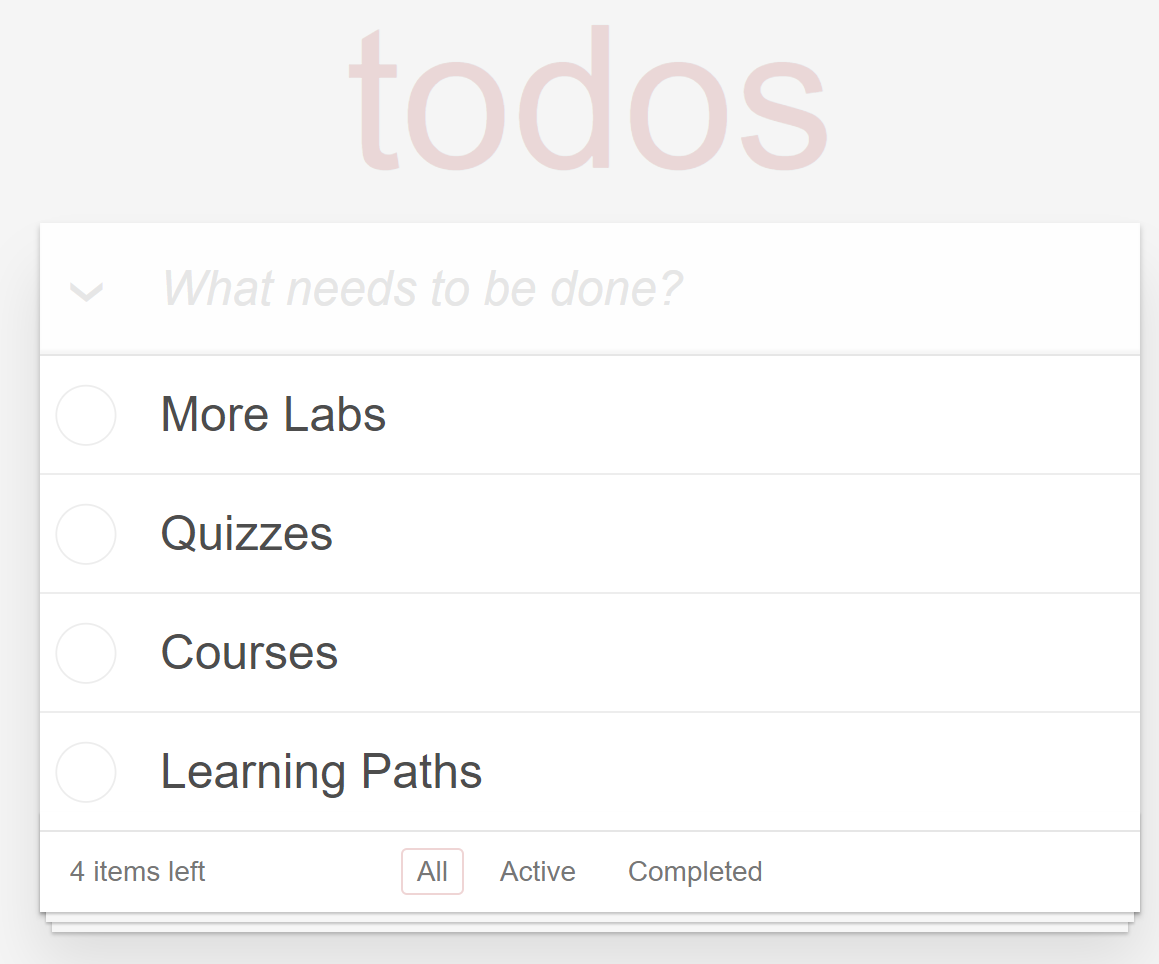


*Note*: You may need to refresh after seeing the **Successfully launched environment: MyApp-env** message to see the **Overview** view.

16. Click on the environment's **URL** to view the application that's been deployed:



17. Enter a few todo notes to see how the application works:



This Lab uses the Angular.js [Todomvc.com](http://todomvc.com) mini-app.

**Summary**

In this Lab Step, you created a Node.js application environment using Elastic BeanStalk. The environment is configured for high availability and to perform rolling updates when new versions of the application are deployed. The application you deployed manages a todo list.

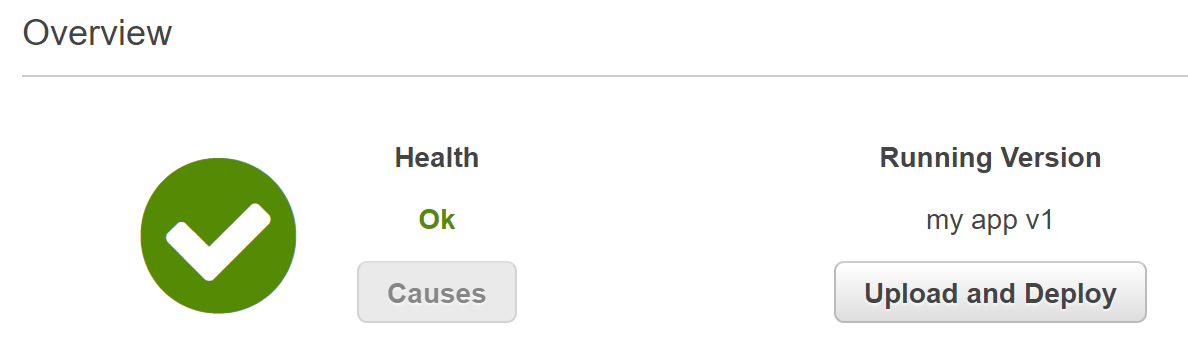
**Running a Rolling Deploy**

**Introduction**

You will deploy a new version of the application using a rolling deployment strategy. This deployment strategy takes a portion of the instances serving traffic out of service to upgrade them to a new version. The instances are returned to serving and a portion of the remaining instances are taken out of service to be upgraded. This process repeats until all instances are serving the new version of the application.

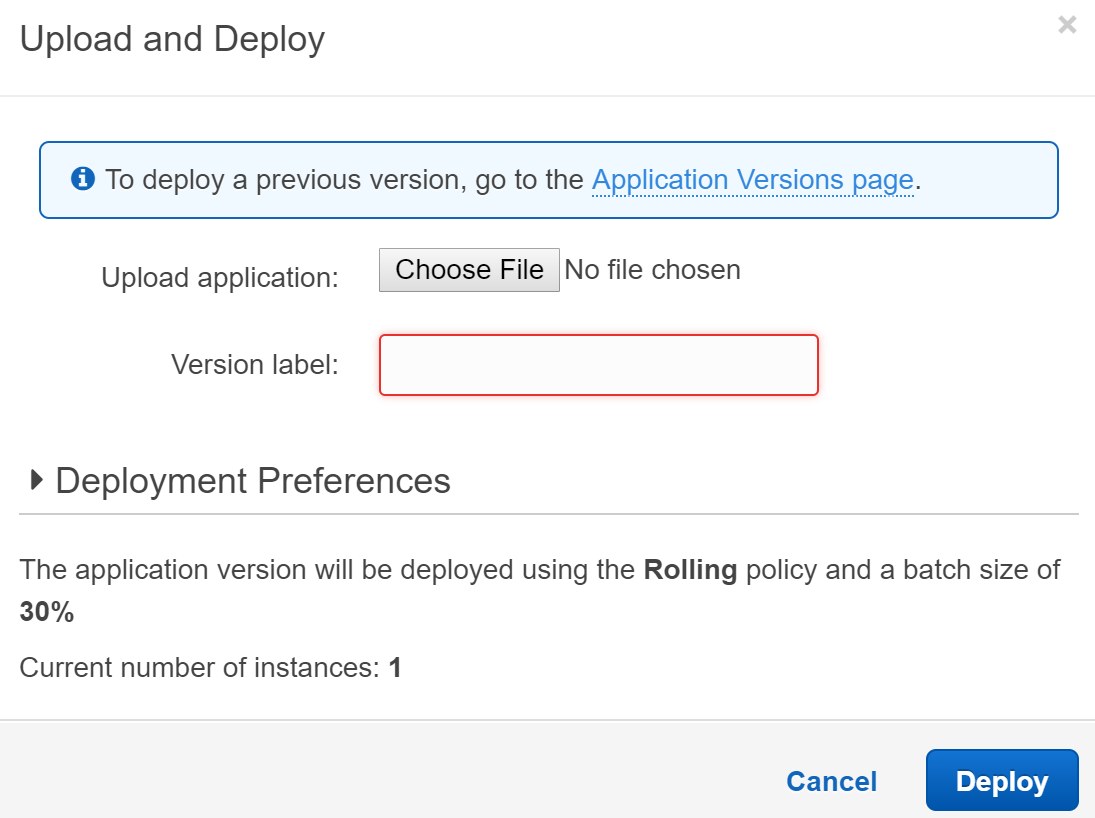
**Instructions**

1. From the Elastic Beanstalk environment **Dashboard**, click on the **Upload and Deploy**button:



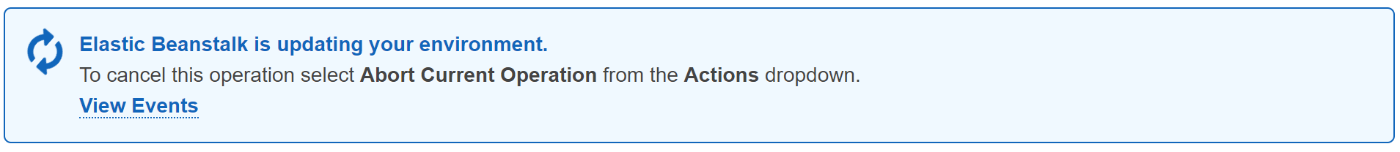
2. In the **Upload and Deploy**form, enter the following values and click **Deploy**:

* **Upload application**: Click **Choose File** and select the application zip file you downloaded at the start of this Lab Step
* **Version label**: *my app v2*



The **Deployment Preferences** use the rolling update configuration you configured when creating the environment. Specifically, using the **Rolling** policy and **30%** batch sizes. You can change the configuration from this form, but the default is what you want. Because only **1**instance is serving the application traffic, the 30% batch size rounds up to the minimum batch size of 1 instance.

After clicking **Deploy**, Elastic Beanstalk will begin working on a mutable rolling deploy of the new application version. The following message is displayed while the deployment is in progress:

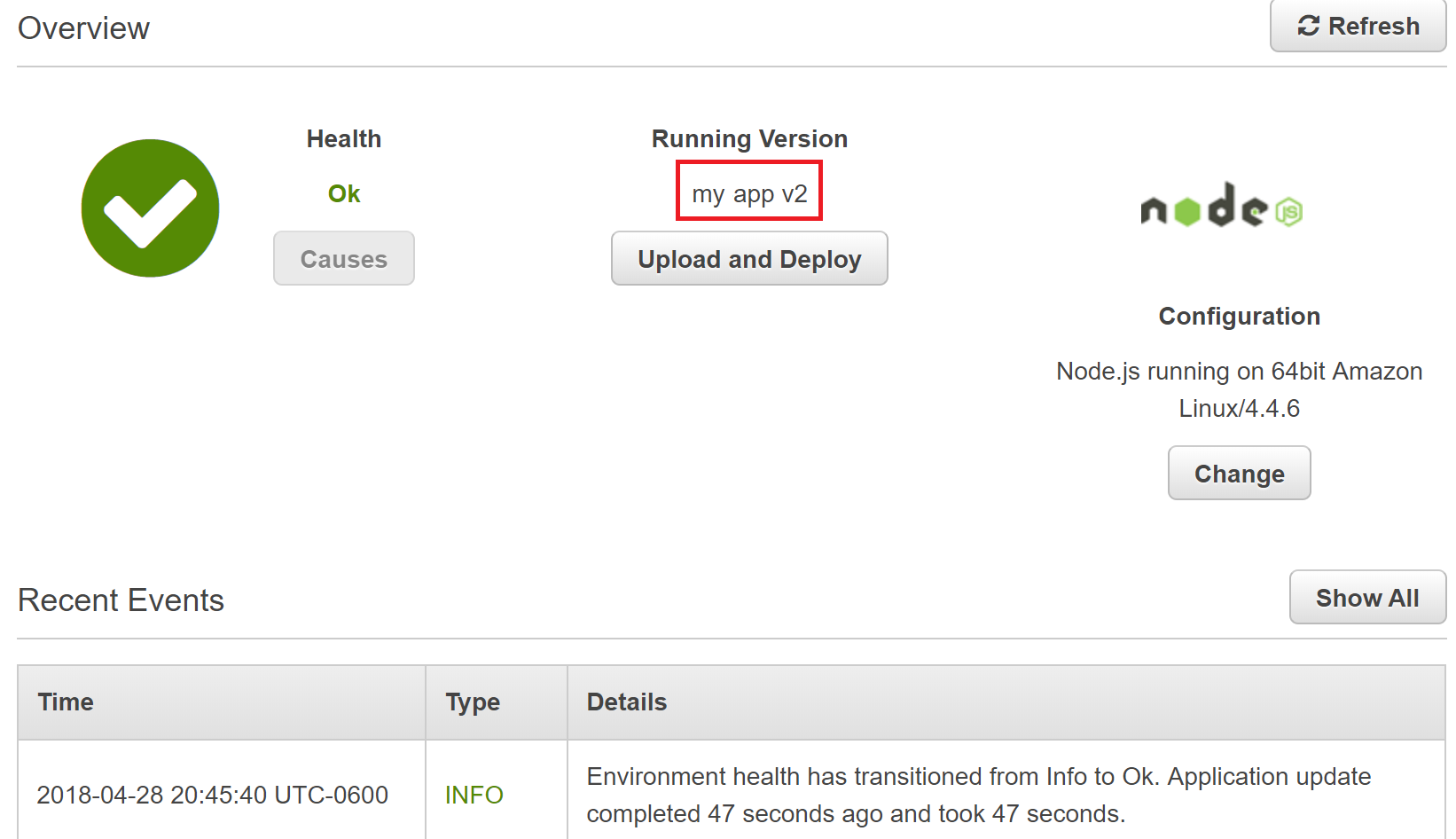


If you refresh the application website during the deployment, you may see the following:

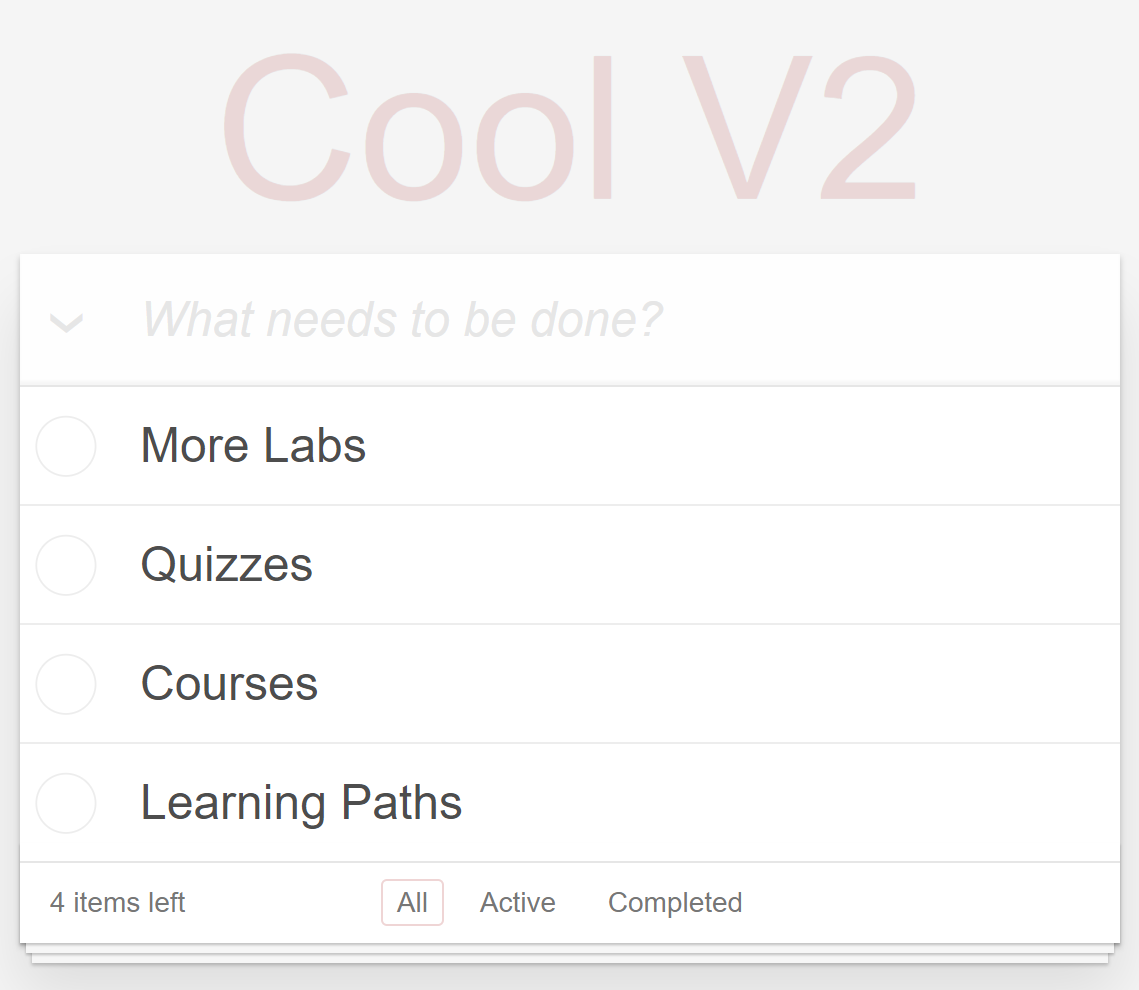


While the instance switches from v1 to v2, the application is temporarily unavailable. With more instances serving the application, it would be possible to avoid an outage but it can't be avoided with only one instance.

3. Wait until the deployment completes, and confirm that your environment reports **v2** is running:



4. Refresh the browser tab that is opened to the application:

[](https://assets.cloudacademy.com/bakery/media/uploads/blobid5-a08873f2-8d6e-4b4c-a550-f5d8ec5fe7fb.png)

The only change made in v2 was altering some text in the todo view from todos to **Cool V2**.

**Summary**

In this Lab Step, you performed a rolling deployment to a new version of the application in Elastic Beanstalk. Some points to keep in mind with the rolling deployment strategy:

* No new instances are created. The new version is deployed in-place on existing instances.
* The maximum available capacity for serving traffic is reduced as instances are taken out of service to be upgraded. This can result in service outages. For example, if only one instance is serving the application.
* When an upgrade is rolled through in more than 2 batches, there will be times when the old and new version of the application can be accessed simultaneously by clients.

**Preparing a Blue-Green Deploy**

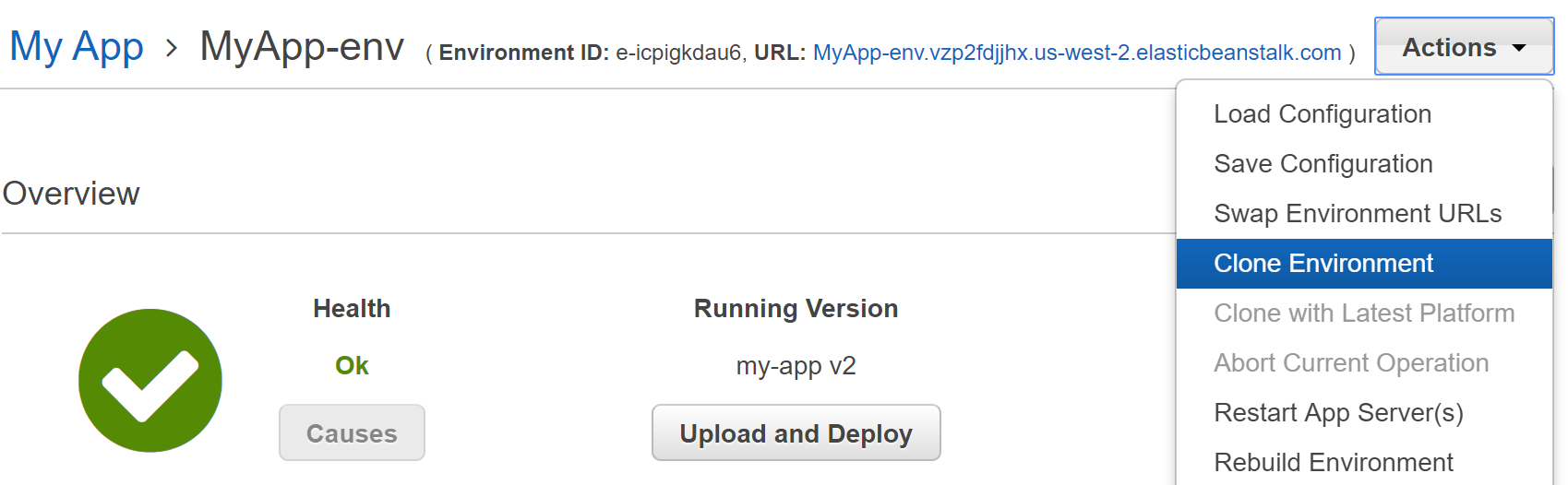
**Introduction**

Now that you have performed a rolling deployment, you will perform another kind of deployment that can overcome some of the shortcomings of rolling deployments. Blue-green deployments work by creating a separate environment to deploy the new version. The original environment, referred to as the *blue* environment, is left untouched while the new environment, referred to as the *green* environment, is spun up running the new version. Once all of the instances in the green environment are serving the new version of the application, new incoming application traffic is sent to the green environment. No new traffic is sent to the blue environment from this point.

You will prepare the green environment in this Lab Step.

**Instructions**

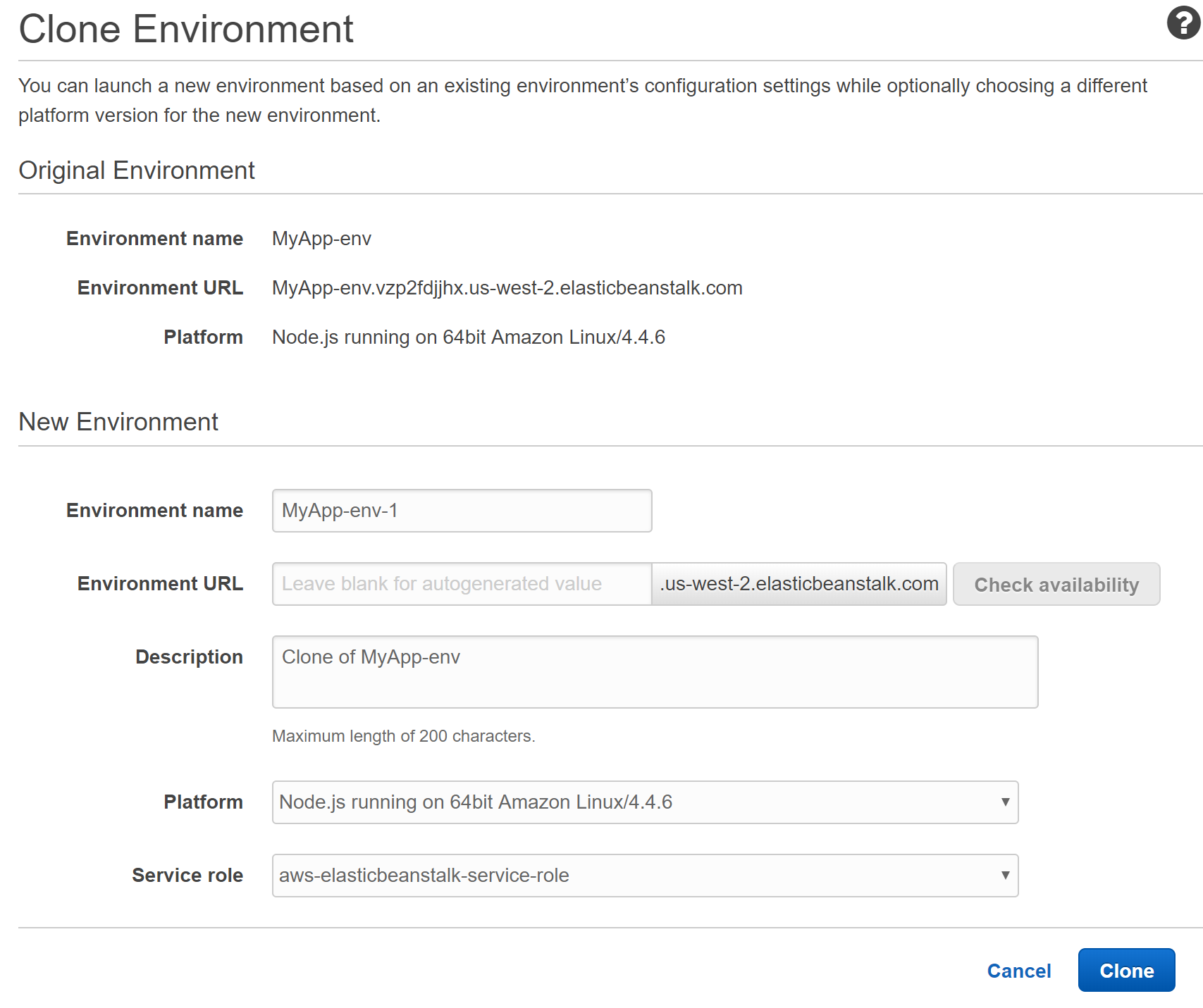
1. From the Elastic Beanstalk environment view, select **Actions** > **Clone Environment** in the upper-right corner to create a second environment that is a clone of the existing environment:



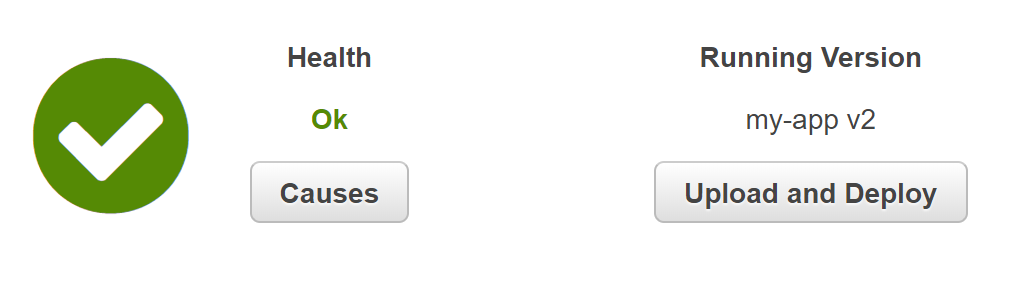
This is an easy way to start configuring a green environment in Elastic Beanstalk.

2. On the **Clone Environment** form, enter the following values leaving the defaults when not specified and click **Clone**:

* **New Environment**:
  + **Environment name**: *MyApp-env-1* (This is the default name. You shouldn't choose a name related to "green" because the green environment becomes the blue environment for the next blue-green deployment)
  + **Environment URL**: Delete the text to have the URL autogenerated



The new environment will take around three minutes to create. Wait until you see the environment **Health**status is **Ok** before proceeding:



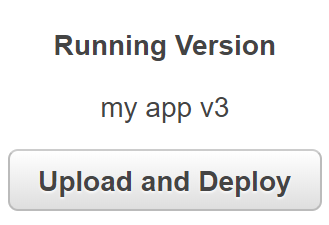
 3. In the Elastic Beanstalk environment **Dashboard**, click the **Upload and Deploy**button.

You will perform a rolling deployment to change the version of the application. Because the environment is unused right now, you don't need to be concerned about the availability and performance concerns surrounding rolling deployments.

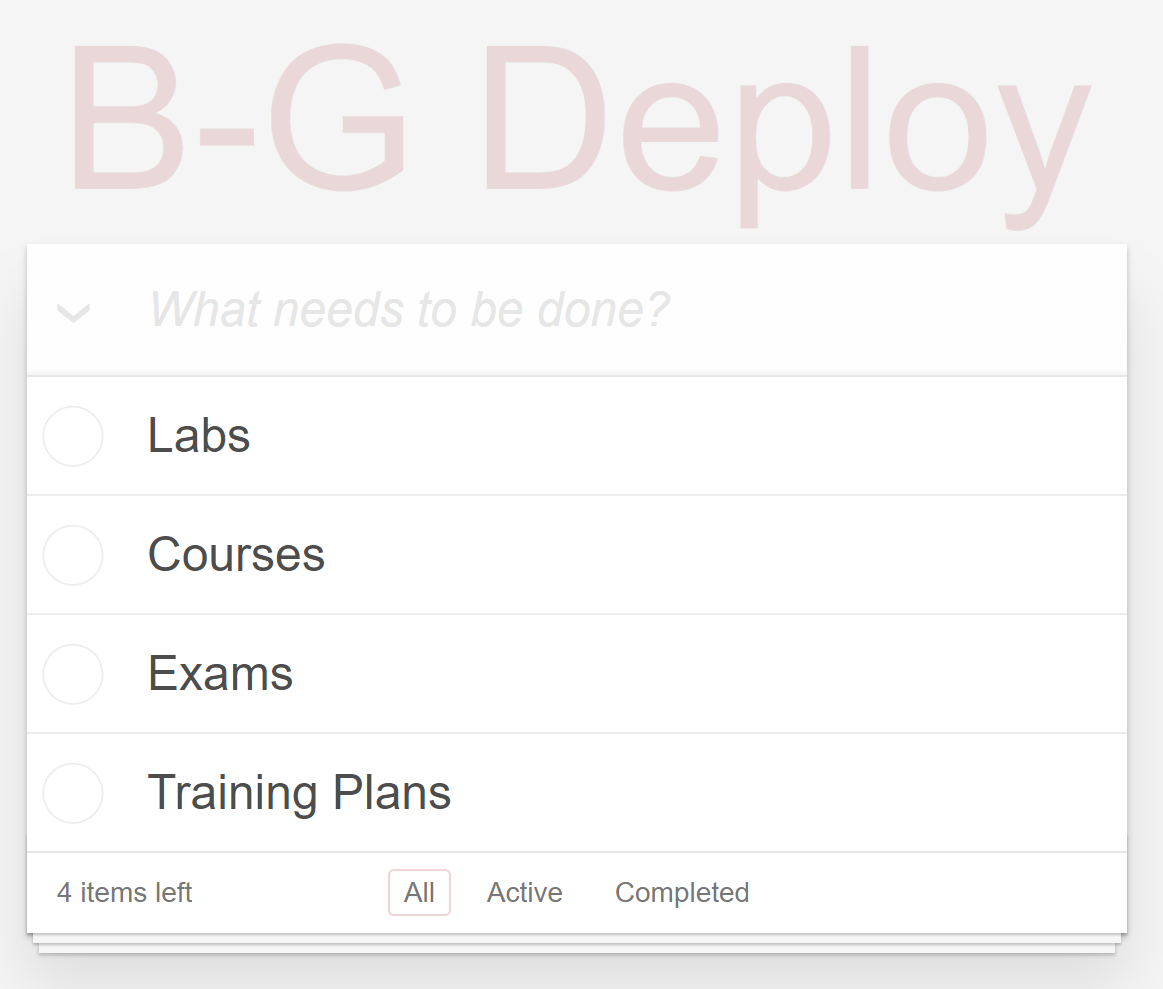
4. In the **Upload and Deploy**form, enter the following values and click **Deploy**:

* **Upload application**: Click **Choose File** and select the application zip file you downloaded in this Lab Step
* **Version label**: *my app v3*

Wait until the rolling deployment completes and you see the **Running Version**is **my app v3**:



5. Click on the environment's **URL** to view the application that's been deployed and verify it is working:



Note that the header text has changed again to **B-G Deploy**.

**Summary**

In this Lab Step, you prepared a green environment by cloning an existing Elastic Beanstalk environment and performing a rolling deployment of the new version of the application. The original blue environment hasn't been modified and is still the active environment.

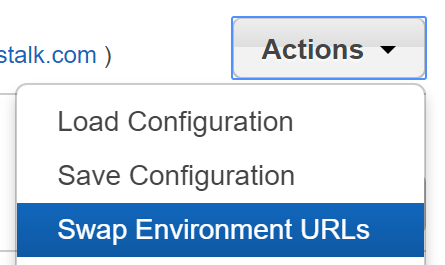
**Doing the Deploy Swap**

**Introduction**

Blue-Green deployments rely on having entire systems or subsystems deployed all at once using a DNS or reverse proxy cutover. You currently have two environments, a green environment that is running v3 and a blue environment that is running v2. The blue environment is using the original URL that is used by clients. In this Lab Step, you will quickly and invisibly swap DNS entries for the two environments, so clients access v3 using the original and more valuable URL. Elastic Beanstalk integrates with Route 53 to accomplish the DNS swap. Route 53 is the DNS management service in AWS.

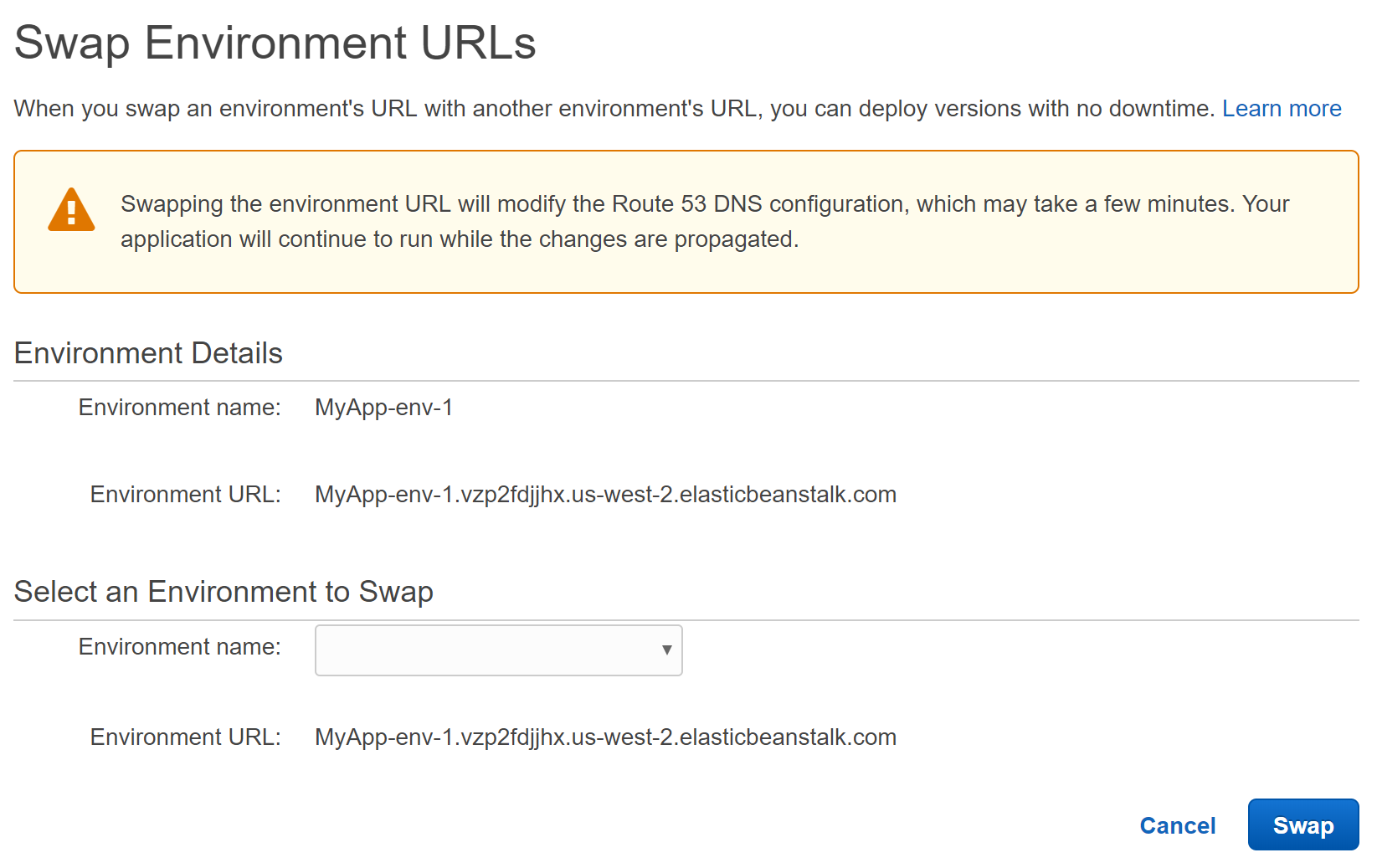
**Instructions**

1. In the new Elastic Beanstalk environment's Console, click on **Actions** > **Swap Environment URLs**:

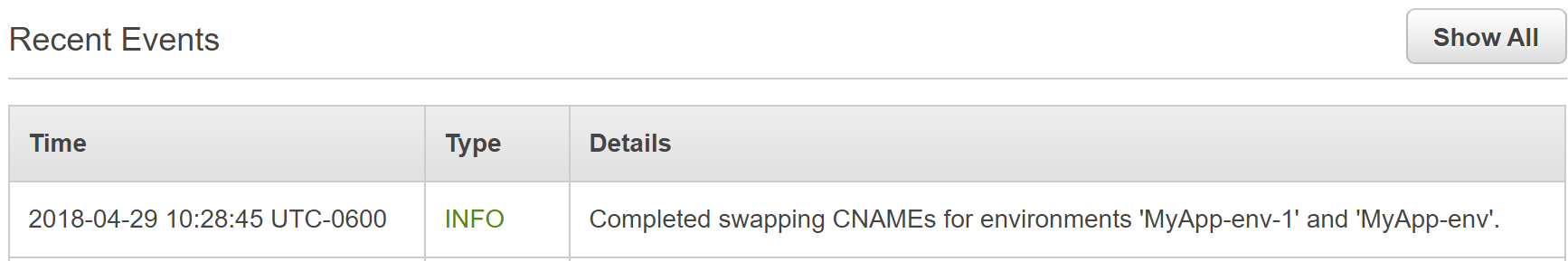
****

 2. In the **Swap Envrionment URLs** form, enter the following values and click **Swap**:

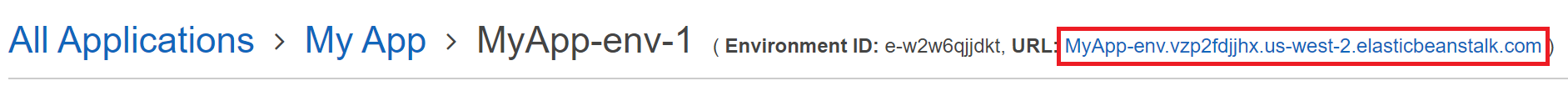
* **Select an Environment to Swap**
  + **Environment name**: MyApp-env



As the warning message mentions, your application continues to run and be accessible to clients while the swap happens. The swap usually completes in under a ten seconds and you will see the **Health** status of **Ok**and the most recent event begins with **Completed swapping CNAMEs**:

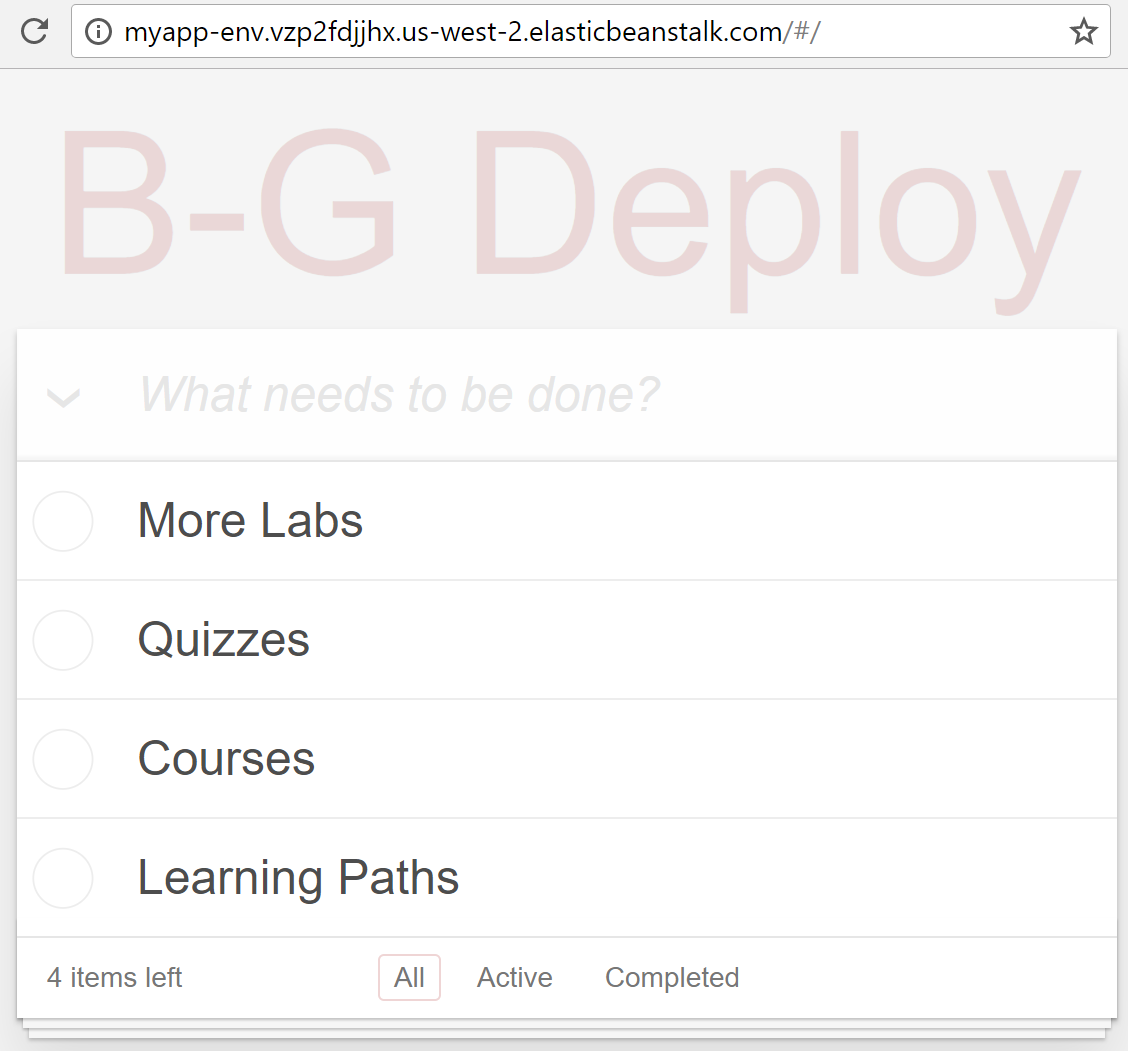


3. Open the **URL**for the **MyApp-env-1** environment:



Notice the URL is the original URL beginning with **MyApp-env.**and not the one beginning with MyApp-env-1.

4. Verify that the browser loads v3 of the app with the **B-G Deploy**header:



If the page still displays the **Cool V2** header, it is because your browser has cached the DNS record that resolves the URL to the original environment's load balancer. Try refreshing the page after a minute until the new version is displayed, or use a different browser. Although there is no downtime with this deployment, the effects of DNS caching are an important consideration when deciding on a deployment strategy.

**Summary**

In this Lab Step, you performed a DNS swap to direct new application traffic to the environment running v3 of the application. This swap is simple to perform and fast. There is no downtime during the cutover since both environments are fully functioning during the process. All new traffic to the original Elastic Beanstalk Environment is now routed to the one running v3, as the v3 environment has taken over all the traffic via DNS.

Some points to keep in mind with the blue-green deployment strategy:

* The maximum number of instances during the deployment is twice the target number of instances. This is because you have two fully functioning environments, one running old version and one running the new version.
* The minimum number of instances serving the application is never below the target number of instances.
* You can easily roll back to the old version because the blue environment is not modified during the deployment.
* The old and new versions of the application are never served simultaneously to new clients.

Elastic Beanstalk performs the cutover switching the blue and green environments at the DNS level in Route 53. Outside of Elastic Beanstalk, blue-green deployments can also perform a cutover at the load balancer level. This is the strategy used by Elastic Container Service (ECS). The issues of DNS caching are avoided because the address of the load balancer remains the same. However, the abrupt cutover doesn't allow connections to gracefully drain.

**Cleaning Up Old Resources**

**Introduction**

Now that you have successfully tried two kinds of low-downtime cloud deployment techniques on Elastic Beanstalk, you might be concerned about the costs associated with these techniques. The deploys are inexpensive. Recall that the rolling deployment runs in-place without creating any new instances so there is no additional cost. The blue-green deployment creates a second environment which potentially doubles the cost while both environments are active. To save costs, you should eventually terminate the environment that was swapped away from during the blue-green deploy. You may choose to wait some amount of time before terminating the environment because:

* Clients may have cached DNS resolutions and still access the old environment for some time
* If you have any doubt about the new issues arising in the new environment, you can quickly swap back to the blue environment to limit the impact

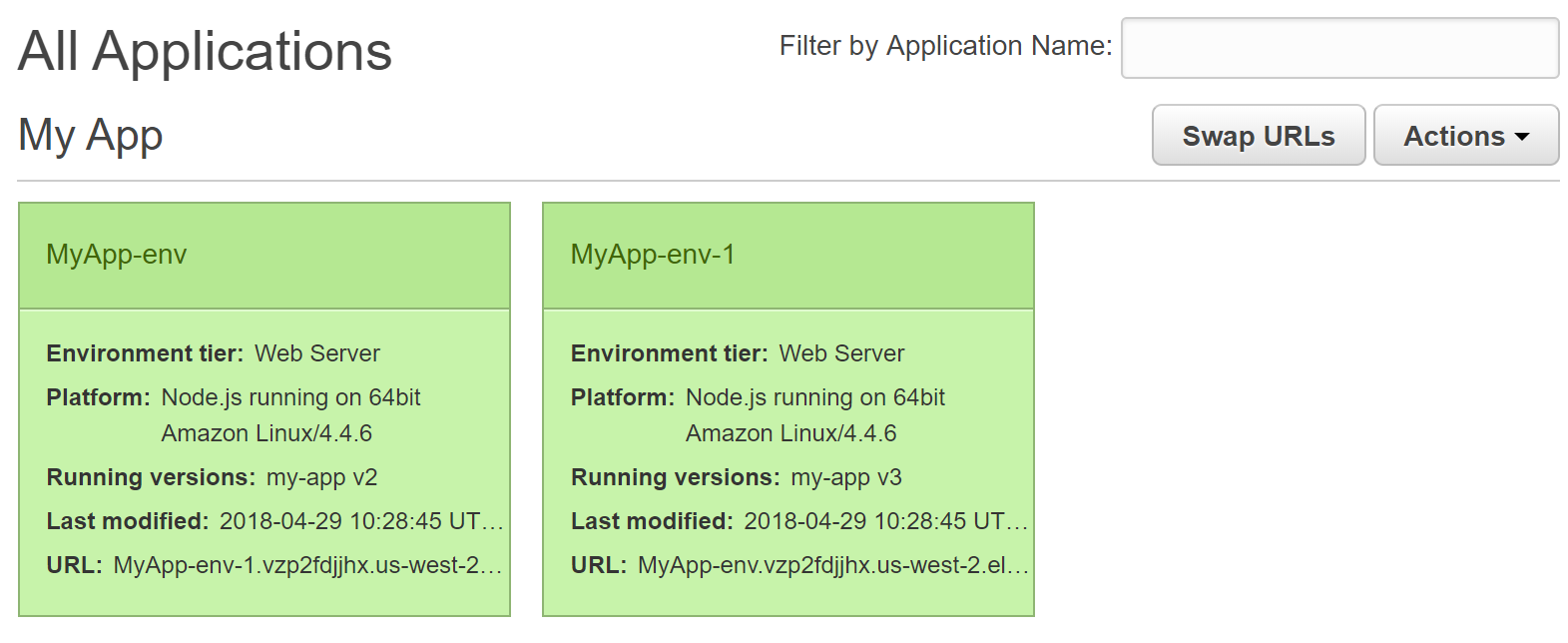
In this Lab Step, you will put saving cost first and terminate the old environment as quickly as possible.

**Instructions**

1. In the new Elastic Beanstalk environment's Console, click **Elastic Beanstalk** the upper-left corner to view all of your applications and environments:



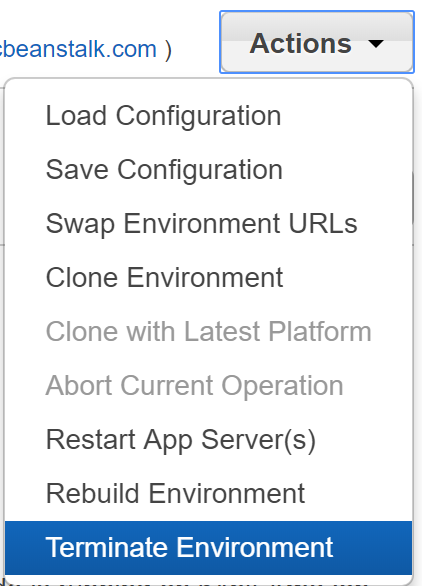
In the **All Applications** view. **My App**is listed as the only application along with two green cards, each representing a live environment associated with the application:



 2. Click on **MyApp-env**, which is currently **Running** **version** **my-app v2**to navigate to the environment's dashboard.

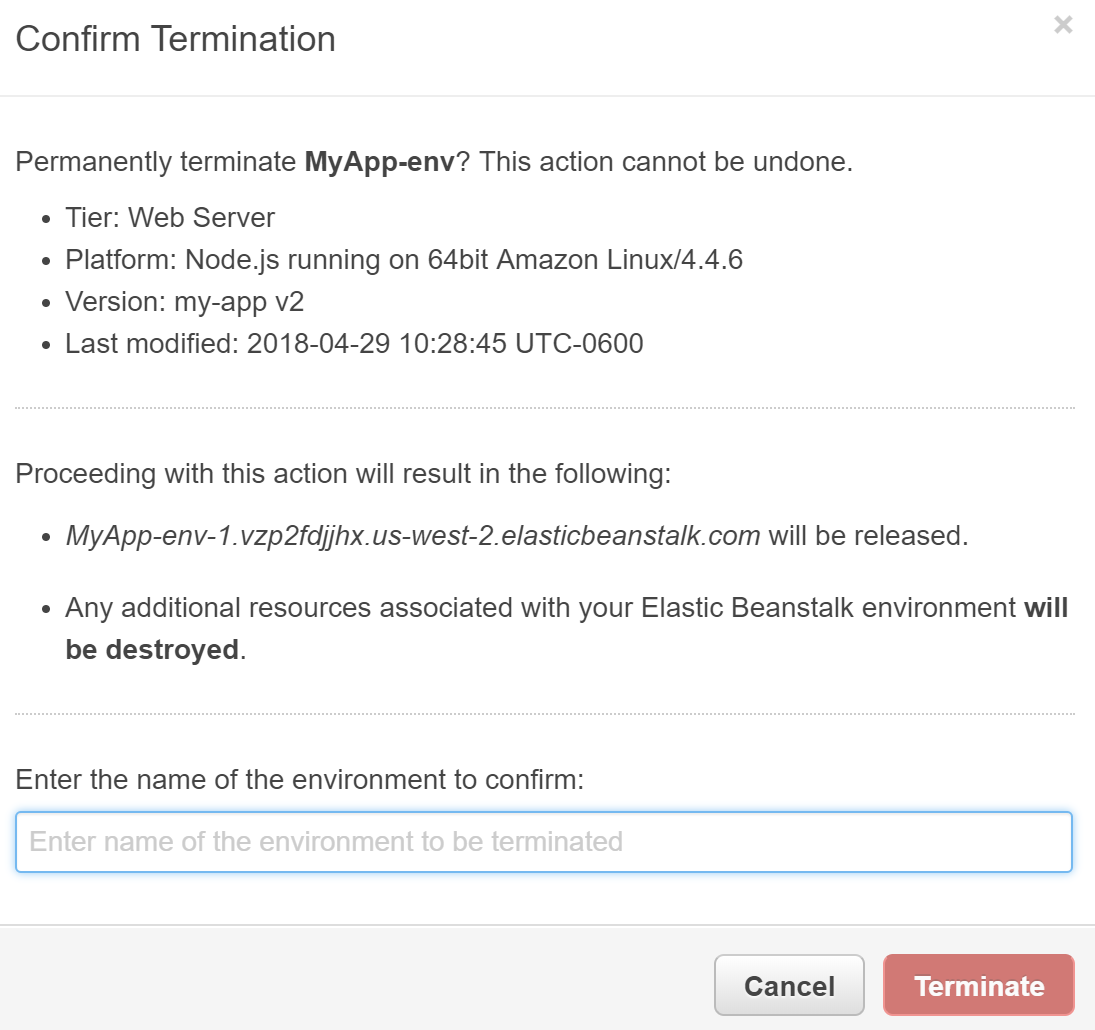
The environment is no longer serving traffic and can be terminated.

3. Click on **Actions** > **Terminate Environment** to destroy the environment:

[](https://assets.cloudacademy.com/bakery/media/uploads/blobid2-c9699b93-3c31-4875-8fce-44ddb010281e.png)

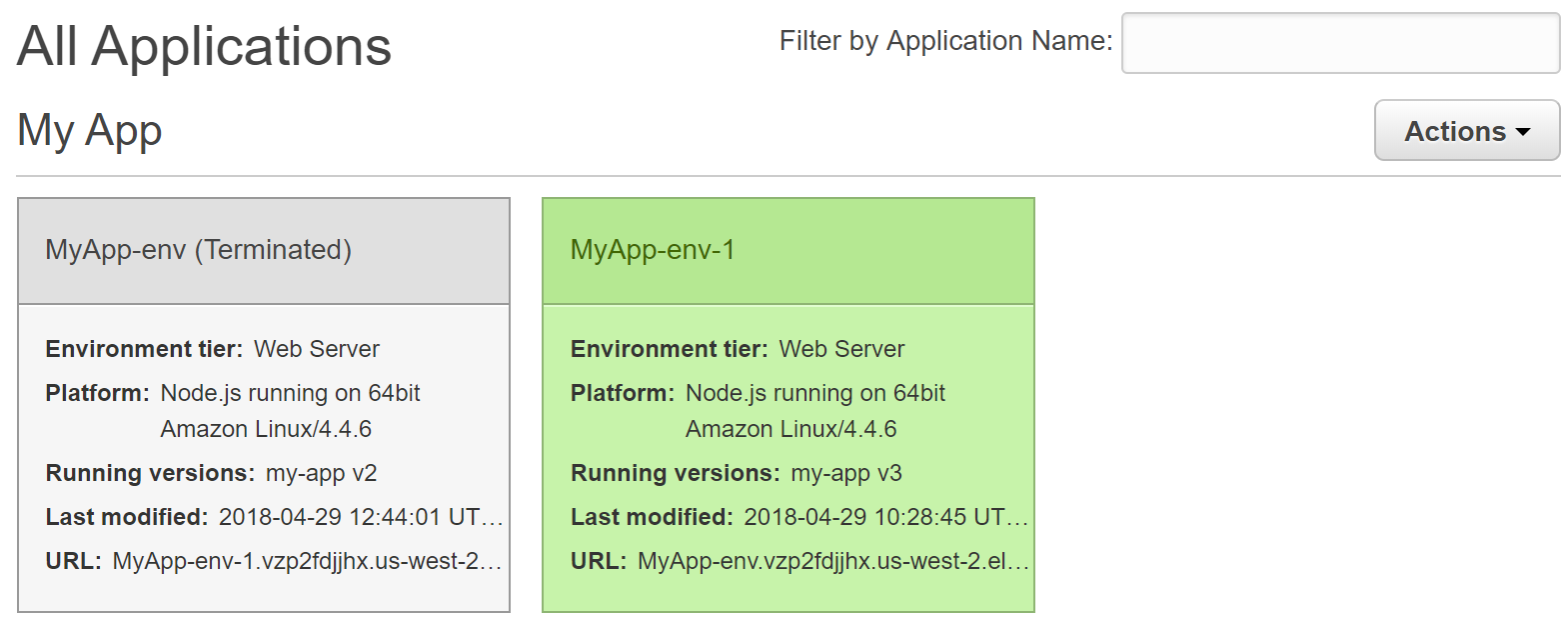
4. In the **Confirm Termination**form, enter the following value and click **Terminate**:

* **Enter the name of the environment to confirm**: *MyApp-env*



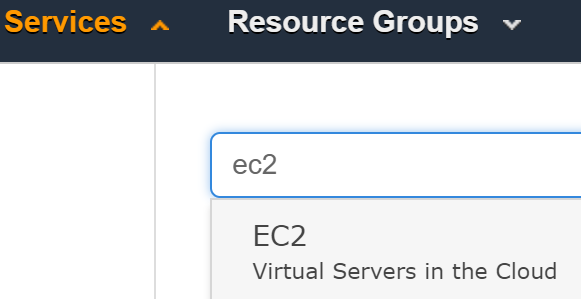
It takes three minutes for all of the resources associated with the environment will be deleted. The **Health** status will transition from **Ok**, to **Severe**(starting when the EC2 instance serving the application is deleted), and finish with **No Data**.

You are returned to the **All Applications** view when the environment is **Terminated**:



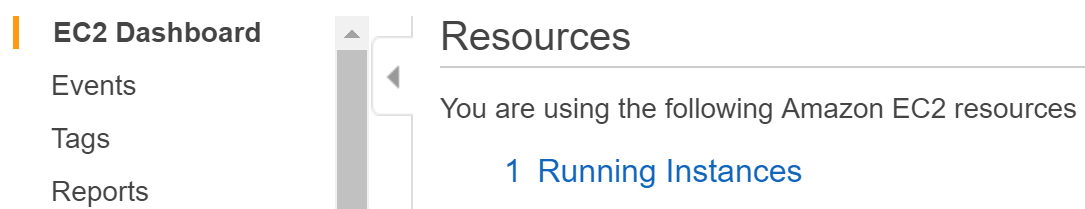
The light gray color indicates a terminated environment. It is possible to restore terminated environments, so they are still listed in the application's list of environments.

 5. Navigate to the AWS Elastic Compute Cloud Console, by clicking on the **Services** in the top-left of the Console, entering *ec2* in the search bar, and clicking on **EC2**:



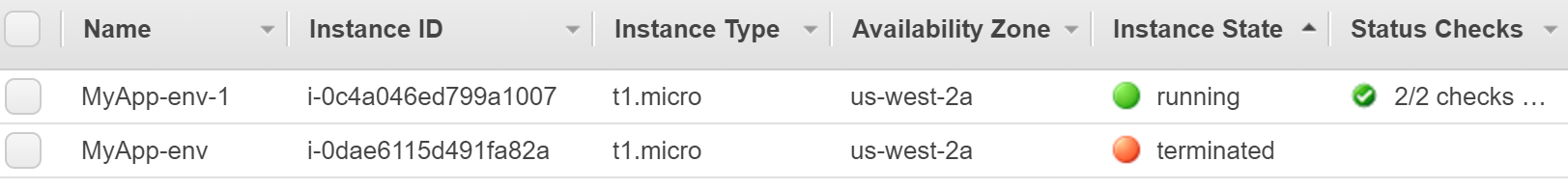
To verify that the spend for the application is minimized you will make sure that the EC2 instance Elastic Beanstalk created for the terminated environment is also terminated.

6. Verify that there is only **1 Running Instance** reported on the **EC2 Dashboard**:



7. Click on **Running Instances** to view a table of instances.

The table includes one **running**, and one **terminated**:



The **terminated** instance is the instance that was associated with the original, swapped out environment named **MyApp-env**.

**Summary**

In this Lab Step, you learned how to terminate an Elastic Beanstalk application environment. You did this to minimize the costs associated with blue-green deployments.