# Deployment in the Cloud with Kubernetes

* To learn about the infrastructure
* Without distributed computing in the cloud we wouldn’t be able to use awesome tools like Spark to make our models run more efficiently

### Introduction

a Kubernetes cluster on **AWS (Amazon Web Services)**. AWS is an on-demand cloud computing service provider. It allows you to purchase and interact with millions of virtual machines in various AWS server farms worldwide. Because of it’s size, AWS can be a complicated tool to use.

**IAM (Identity and Access Management) and EC2 (Elastic Compute 2)**. IAM allows you to manage access to the AWS resources that you use. You can set different permissions on certain things. EC2 provides scalable computing capacity in the AWS cloud. It lets you launch as many virtual machines as you need.

* Loading our public **AMI (Amazon Machine Image)**
* Setting up IAM roles for use with EC2 instances
* Setting up *security groups* so that we can access our instances
* Create *launch configurations* to set up our instances
* Create *auto scaling groups* to spin up our instances
* Deploy Docker images to Kubernetes

**Note: Everything ON AWS with be through the us-east-1 (sometimes referred to as US East (N. Virginia)).**

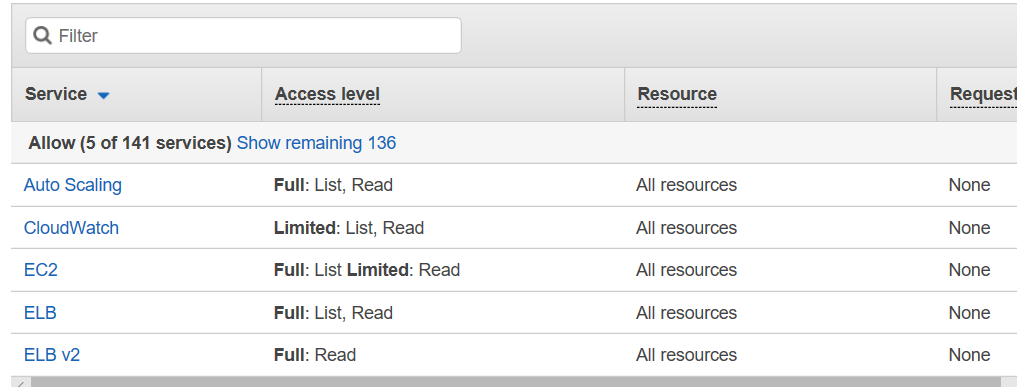
## The Setup

### 1. IAM Steps

We’re going to start by creating a new IAM (Identity and Access Management) role.

1. Go to Services (top tab) > IAM (under Security, Identity & Compliance) > Roles (left bar)
2. Click “Create New Role” (blue button) > Click “Select” on “Amazon EC2”
3. Filter for “AmazonEC2ReadOnlyAccess”. Check the box and click “Next Step” (blue button)

AmazonEC2ReadOnlyAccess will provides read only access to Amazon EC2 via the AWS Management Console.



{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": "ec2:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "elasticloadbalancing:Describe\*",

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": [

"cloudwatch:ListMetrics",

"cloudwatch:GetMetricStatistics",

"cloudwatch:Describe\*"

],

"Resource": "\*"

},

{

"Effect": "Allow",

"Action": "autoscaling:Describe\*",

"Resource": "\*"

}

]

}

1. Role Name will be “joyce\_keburnetes”
2. Create the role (blue button)

tasks:

- name: create IAM role

iam\_role:

name: joyce\_keburnetes

state: present

assume\_role\_policy\_document: "{{ lookup('file','policy.json') }}"

managed\_policy:

- arn:aws:iam::aws:policy/AmazonEC2ReadOnlyAccess

- name: wait a bit for the IAM role to persist

pause:

seconds: 15

### 2.EC2 Intro

1. Go to Services (top tab) > EC2

### 3. EC2 Security

It’s time to set up how we can access our cluster from the outside world.

1. Under “Network & Security” (left bar) click “Security Groups”
2. Click “Create Security Group” (blue button)
3. Security Group name is “joyce-kubernetes”
4. Put in any description and click “Create” (blue button)
5. Click on the security group that you just created
6. Copy the Group Id from the second column of the security group
7. Click on “Inbound” in the bottom tab and click edit
8. Select “All Traffic” from the first dropdown and paste your Group Id in the last box
9. Click “Add Rule”. Select “Custom TCP Rule” from the first box. Give it a port range of “30000 - 30100”. Select “My IP” from the last dropdown.

* Note: These are the ports that we’ll use to access our application from the Kubernetes cluster

1. Click “Add Rule”. Select “Custom TCP Rule” from the first box. Give it a port range of just “22”. Select “My IP” from the last dropdown.
2. Click “Save”

- name: create security group

ec2\_group:

region: "us-east-1"

name: joyce-kubernetes

description: joyce-kubernetes

state: present

rules:

- proto: all

group\_name: joyce-kubernetes

- proto: TCP

from\_port: 22

to\_port: 22

cidr\_ip: "{{ local\_ip\_address }}/32"

- proto: TCP

from\_port: 30000

to\_port: 30100

cidr\_ip: "{{ local\_ip\_address }}/32"

rules\_egress:

- proto: all

cidr\_ip: 0.0.0.0/0

### 4. EC2 SSH Key Pairs

You need to create an ssh key pair which will allow you to access your instances on AWS so you can set them up

1. Network & Security (left bar) click Key Pairs
2. Click “Create Key Pair” (blue button) and call it “joyce-kubernetes”

* Note: It will download it for you
* This key will need to be shared with anyone accessing the cluster so keep it safe

name: create key pair

ec2\_key:

region: "us-east-1"

name: joyce-kubernetes

state: present

key\_material: "{{ item }}"

with\_file: id\_joyce.pub

### 5. EC2 Master Launch Configuration

The launch configuration is the template for the master auto-scaling group

1. Under “Auto Scaling” (left bar) click “Launch Configurations”
2. Click “Create Auto Scaling Group” (blue button)
3. Click “Create Launch Configuration” again
4. Click “Community AMIs” (left bar)
5. Search for “ami-e8c1e793”
6. Select it
7. Select “m4.large” > Click next
8. Name it: “joyce-kubernetes-master”, select the “joyce-kubernetes” IAM, and click “Next” **2 times**
9. Click the radio button that says “Select an existing security group” and select the security group that we made called “joyce-kubernetes”
10. Click review and click “Create Launch Configuration”
11. “Choose existing key pair” and selected the one we made called “joyce-kubernetes”. Click “Create Launch Configuration”

### 6. EC2 Master Auto Scaling Group

The auto scaling groups will be used to create the instances and control how many VMs we have and scale them to production demands.

1. Under “Auto Scaling” (left bar) click “Auto Scaling Groups”
2. Make sure that your launch configuration is set to “kubernetes-master”
3. Set your group name also to “kubernetes-master”
4. Set group size to start with **0** instances
5. Select the Subnet that ends with “us-east-1a”

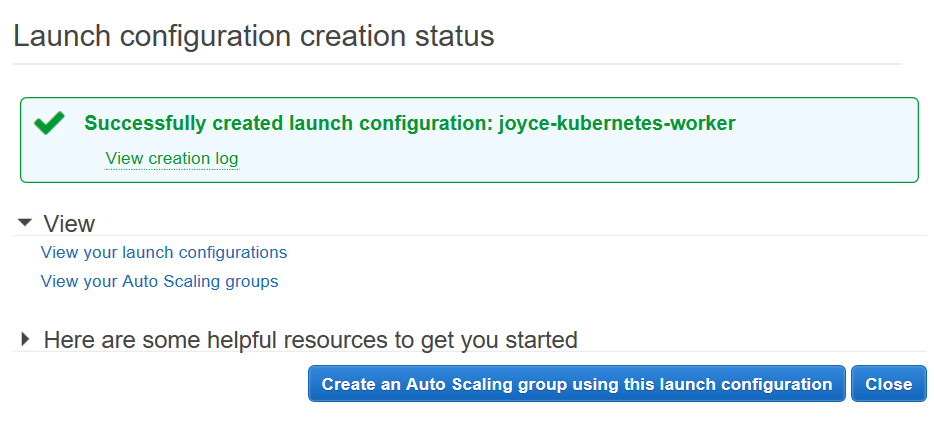
* **This is critical.** If you don’t do this you will be sad very soon

1. Click “Next” **3 times**
2. Set the Key to “Name” (**casing is important here**) and the value to “joyce-kubernetes-master”
3. Click “Add Tag” and set the Key to “Role” (**casing is important here**) and the value to “master”
4. Click “Create Auto Scaling Group”

### 7. EC2 Worker Launch Configuration

The launch configuration is the template for the worker auto-scaling group

1. “Autoscaling” (left bar) click “Launch Configurations”
2. Click “Create Auto Scaling Group” (blue button)
3. Click “Create Launch Configuration” again
4. Click “Community AMIs” (left bar)
5. Search for “ami-e8c1e793”
6. Select it
7. Select “c4.2xlarge” > Click next
8. Name it: “joyce-kubernetes-worker”, select the “joyce\_kubernetesr” IAM, and click “Next” **2 times**
9. Click the radio button that says “Select an existing security group” and select the security group that we made called “joyce-kubernetes”
10. Click review and click “Create Launch Configuration”
11. “Choose existing key pair” and selected the one we made called “kubernetes”. Click “Create Launch Configuration”



Click “Create an Auto Scaling group using this launch configuration”

### 8. EC2 Worker Auto Scaling Group

The auto scaling groups will be used to create the instances and control how many VMs we have and scale them to production demands.

1. Make sure that your launch configuration is set to “joyce-kubernetes-worker”
2. Set you group name also to “joyce-kubernetes-worker”
3. Set group size to start with **0** instances
4. Select the Subnet that ends with “us-east-1a”

* **This is critical. If you don’t do this you will be sad very soon**

1. Click Next 3 times
2. Set the Key to “Name” (**casing is important here**) and the value to “joyce-kubernetes-worker”
3. Click “Add Tag” and set the Key to “Role” (**casing is important here**) and the value to “worker”
4. Click “Create Auto Scaling Group”

### 9. EC2 Creating the Master Instance

1. “Auto Scaling” (left bar) click “Auto Scaling Groups”
2. Select the “joyce-kubernetes-master” auto scaling group we created
3. Click “Edit” in the details tab
4. Set “Desired” to 1 and “Max” to 1 and click “save”

### 10. EC2 Creating the Worker Instance

1. “Auto Scaling” (left bar) click “Auto Scaling Groups”
2. Select the “joyce-kubernetes-worker” auto scaling group we created
3. Click “Edit” in the details tab
4. Set “Desired” to 1 and “Max” to 1 and click “save”

### 11. EC2 Looking up Instances

Now we’re going to create instances with the master and worker templates that we set up above.

1. Go to the Instances (left bar) and click “Instances”
2. Click the gear icon in the top right corner and make sure that “Role” is checked on left side and click “close”
3. Wait for a bit to make sure that the that your instances pop up
4. Click the “joyce-kubernetes-master” instances and select it. Copy the IPv4 Public IP Address from the “Details” tab below

### 12. SSHing into our Instances

Let’s step back from the AWS web interface and actually get into the instances that we created.

* Windows Users

1. Open PuttyGen
2. Click “Load Private Key” and browse from the private key that you downloaded in step 4B

* In the file format drop down click “All Formats” and browse for the file that you downloaded

1. Click “Save Private Key” and don’t set a password
2. Open Superputty
3. Go to “Tools” (in the menu) and select “Putty Config”
4. Go to Connection > SSH > Auth
5. Browse for the private key and select the one you created in step 12B
6. Click the session section and select “default” from the configuration box
7. Click Save and then click cancel
8. Top left bar under host,put the IP address copied in Step 11D
9. Under “Login As” type “ubuntu”
10. Click “Connect”

* It will ask you to confirm the identity of the host. Click “yes”.

1. Click Save and click cancel

* OSX and Linux

1. Open your terminal
2. Type `chmod 600 <PATH-TO-FILE-DOWNLOADED-IN-4B>`
3. Type `ssh -i <PATH-TO-FILE-DOWNLOADED-IN-4B> ubuntu@<IP-ADDRESS-COPIED-IN-STEP-11D>`

### 13. Deploying the Kubernetes Dashboard

Finally! Let’s deploy the reason that we went through all of this trouble.

1. Become root by typing `sudo su`
2. Type `export KUBECONFIG=/etc/kubernetes/admin.conf`

* You will have to type this in every time you log in to the machine
* If this file does not exist, your kubernetes cluster might not be ready yet. Check again in a minute or so

1. Type `kubectl create -f /etc/zeppelin/kubernetes/canal.yaml`

* This, like the previous meetup, will install a pod network into your cluster

1. Type `kubectl create -f /etc/zeppelin/kubernetes/dashboard.yaml`

* After a while you will be able to use the IP copied from step 11D with port **30000** to access the website based kubernetes dashboard from your browser!!

Note:

* It may take a while before kubernetes is ready. You can type `kubectl get nodes` to see if kubernetes has started. If it has not started, you will see at least one node with the state ”NotReady”.
* Once you have done step 13C the nodes states will change to “Ready”.
* It may take a while for the dashboard to become ready. In order to see the status of everything in kubernetes you will have to type: `kubectl get po --all-namespaces -o wide`. When it’s ready the status of every item should be “Running”

### 14. Deploying our Spark/TensorFlow Web Application (optional)

1. Type `kubectl create -f /etc/zeppelin/kubernetes/zeppelin.yaml`
2. Type `kubectl create -f /etc/zeppelin/kubernetes/spark-master.yaml`
3. Type `kubectl create -f /etc/zeppelin/kubernetes/spark-slave.yaml`

* After a while you will be able to use the IP copied from step 11D with port **30001** to access the zeppelin from your browser!
* If, for some reason, you aren’t able to access the zeppelin application, it may be because it is running on the worker Amazon Instance. In order to access the application you will have to get the IP address of the worker from the AWS Instances page like in step 11D.

### 15. Don’t Throw Moneys Away (Clean up)

**If you don’t do this step AWS could end up charging you a whole lot of money. Let’s take down our instances now.**

1. Under “Auto Scaling Groups” (left bar under Auto Scaling)
2. Select the “kubernetes-master” auto scaling group created in step 6I
3. Click “Edit” (Under the Details section)
4. Set both “Max” and “Desired” to 0

* If you started with one instance when creating the group you will also need to set “Min” to 0

1. Wait for the instances column to go to 0. After this is done you can select the auto scaling group and then right click and then select “Delete”
2. Repeat steps 15A through 15E for the auto scaling group created in step 8
3. Go to the “Launch Configurations Page”
4. Select **both** launch configurations. Right click them and “Delete”
5. Go to the “Key Pairs” page and delete the key pair that we created in step 4B
6. Go to the “Security Group” page and delete the one you created in step 3D
7. Go to the “IAM Role” page and delete the one you created in step 1E
8. As a sanity check you can look at the instances page to make sure that your instances and terminated and that you are not spending any money