**AWS — Deploying Angular App With Java Backend On EKS**

A step by step guide with an example project



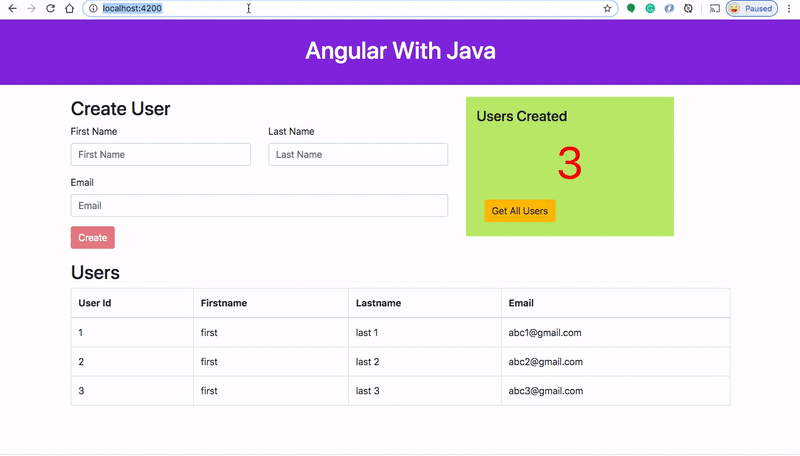
AWS provides more than 100 services and it’s very important to know which service you should select for your needs. Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane. Kubernetes is an open-source system for automating the deployment, scaling, and management of containerized applications.

In this post, we are going to deploy the Angular application with the Java environment. First, we dockerize our app and push that image to Amazon ECR and run that app on Amazon EKS.

* ***Example Project***
* ***Prerequisites***
* ***Dockerize the Project***
* ***Pushing Docker Image To ECR***
* ***Create a Cluster and Worker Nodes***
* ***Configure kubectl to use Cluster***
* ***Deploy Kubernetes Objects On AWS EKS Cluster***
* ***Summary***
* ***Conclusion***

***Example Project***

This is a simple project which demonstrates developing and running Angular application with Java. We have a simple app in which we can add users, count, and display them at the side, and retrieve them whenever you want.



**Example Project**

If you want to practice your own here is a Github link to this project. You can clone it and run it on your machine as well.

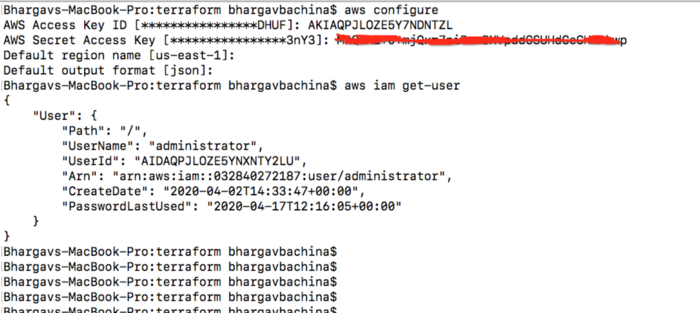
//clone the project  
git clone <https://github.com/bbachi/angular-java-ecs.git>// Run Angular on port 4200cd /src/main/ui  
npm install  
npm start// Run Java Code on 8080  
mvn clean install  
java -jar target/users-0.0.1-SNAPSHOT.jar

**Prerequisites**

* If you are new to Angular please go through the below link on how to develop and build the Angular project with Java backend.

[How To Develop and Build Angular App With Java Backend](https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-angular-app-with-java-backend-87fb603c6e17)

* AWS account setup: AWS offers a free tier for one year [here is the link to set it up.](https://portal.aws.amazon.com/billing/signup?redirect_url=https%3A%2F%2Faws.amazon.com%2Fregistration-confirmation#/start)
* Once you set it up you have a root account. It’s not a best practice to use your root account to do any tasks instead you should create an IAM group that has permissions for administrator access and add a user to it and log in with that user.
* [Install AWS CLI](https://aws.amazon.com/cli/)
* Configure AWS CLI for the user you just created above. You should use this command aws configure and it will ask for access key id and secret key.



**log in with user credentials**

* You have to install Docker for Desktop (whatever your OS is). [Please follow this link](https://docs.docker.com/install/)to install Docker on your laptop. Once installed you can check the Docker info or version with the following commands.

docker info  
docker --version

* The Kubernetes command-line tool, [kubectl](https://kubernetes.io/docs/reference/kubectl/kubectl/), allows you to run commands against Kubernetes clusters. [Install it from here.](https://kubernetes.io/docs/tasks/tools/install-kubectl/)

**Dockerize the Project**

Amazon EKS is a managed service that makes it easy for you to run Kubernetes on AWS. The first thing you need to do is to dockerize your project.

Here is the Dockerfile and it is using openjdk as a base image.

**Dockerfile**

If you want to know the whole process of creating a Docker image for this project, please check the below link.

[Dockerizing Angular App With Java Backend](https://blog.bitsrc.io/dockerizing-angular-app-with-java-backend-fedb96919bc9)

Here are the commands to build the image and run it on the Docker engine on your local machine.

// build the image  
docker build -t java-angular .// run the container  
docker run -d -p 8080:8080 --name javaangular java-angular// list the container  
docker ps// logs  
docker logs javaangular

**Pushing Docker Image To ECR**

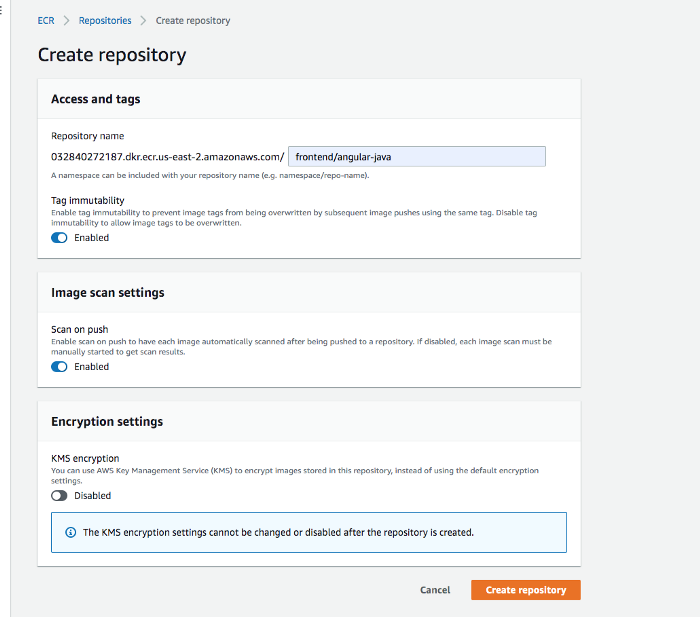
Amazon Elastic Container Registry (ECR) is a fully-managed [Docker](https://aws.amazon.com/docker/) container registry that makes it easy for developers to store, manage, and deploy Docker container images. Amazon ECR is integrated with [Amazon Elastic Container Service (ECS)](https://aws.amazon.com/ecs/), simplifying your development to production workflow.

Amazon EKS works with any Docker registry such as Docker Hub, etc. But, in this post, we see how we can use Amazon ECR to store our Docker images. Once you set up the Amazon account and create an IAM user with Administrator access the first thing you need to create a Docker repository.

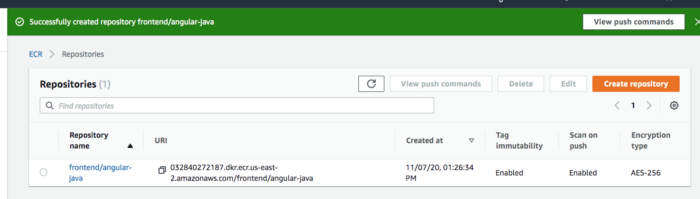
You can create your first repository either by AWS console or AWS CLI

**AWS console**

Creating a repository with AWS console is straightforward and all you need to give a name.



**creating repository**



**repository**

**AWS CLI**

The first thing you need to do is authenticate to your default registry. Here is the command to authenticate to your default registry. You need to make sure you are putting the correct regions and account id in the command.

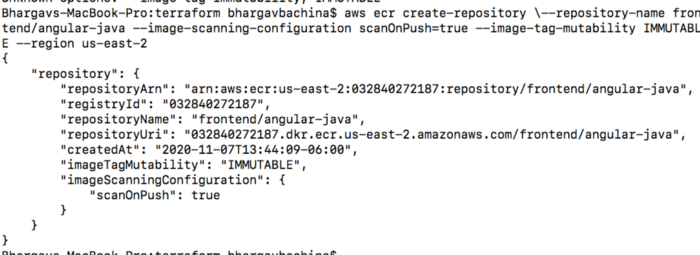
aws ecr get-login-password **--region us-east-1** | docker login --username AWS --password-stdin **aws\_account\_id**.dkr.ecr.**us-east-1**.amazonaws.com



**Authenticating to ECR**

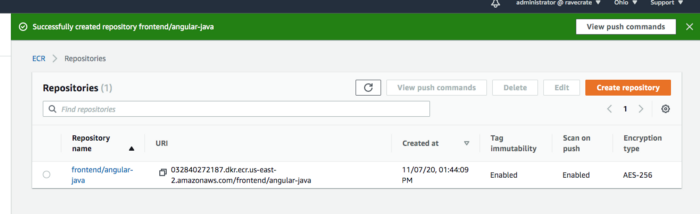
It’s time to create a repository with the following command

**aws ecr create-repository \   
 --repository-name frontend/angular-java\   
 --image-scanning-configuration scanOnPush=***true* **\  
 --image-tag-mutability IMMUTABLE   
 --region** *us-east-2*



**Creating repository**

You will have the same result like this as well.



**Repository**

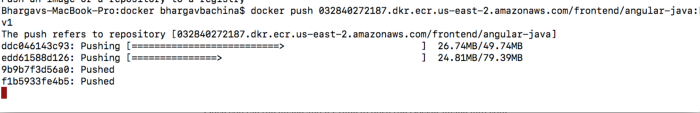
**Tagging your local Docker image and Pushing**

You have created a Docker image on your local machine earlier. It’s time to tag that image with this repository URI in the above image.

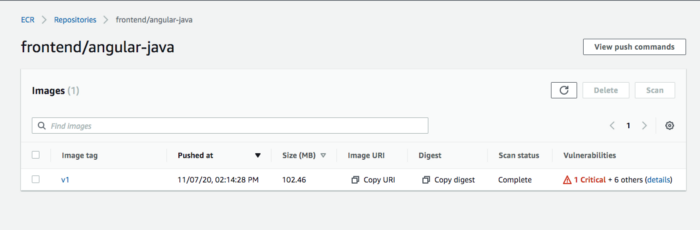
docker tag java-angular:latest 032840272187.dkr.ecr.us-east-2.amazonaws.com/frontend/angular-java:v1

Once you tag the image and it’s time to push the Docker image into your repository.

// list the images  
docker images// push the image  
docker push 032840272187.dkr.ecr.us-east-2.amazonaws.com/frontend/angular-java:v1



**pushing Docker image**



**Docker image with tag v1**

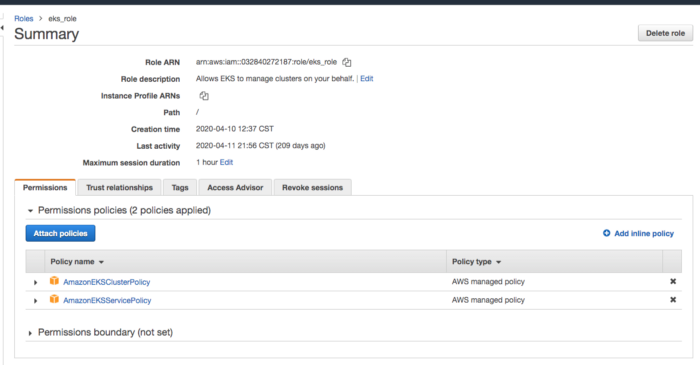
**Create a Cluster and Worker Nodes**

Getting started with AWS EKS is easy all you need to do the following steps

* We need to create an AWS EKS cluster with AWS console, SDK, or AWS CLI.
* Create a worker node group that registers with EKS Cluster
* When your cluster is ready, you can configure **kubectl** to communicate with your cluster.
* Deploy and manage your applications on the cluster

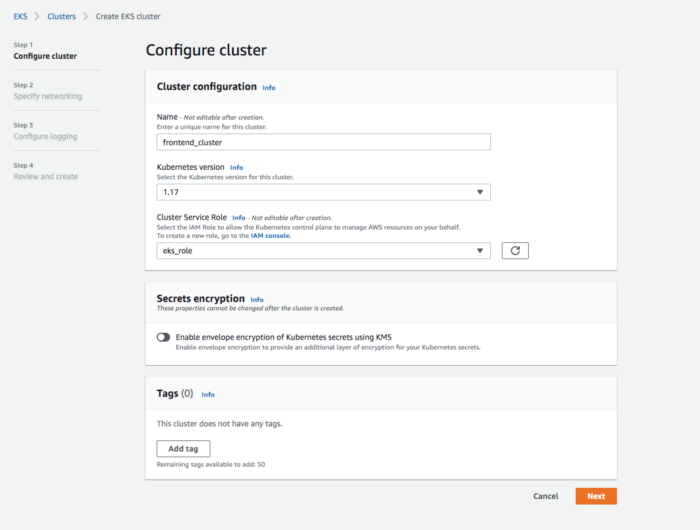
**Cluster Creation**

Let’s create a cluster by [following this guide here](https://docs.aws.amazon.com/eks/latest/userguide/create-cluster.html). Make sure you created a role for the EKS to allow Amazon EKS and the Kubernetes control plane to manage AWS resources on your behalf. I created a role called **eks\_role**



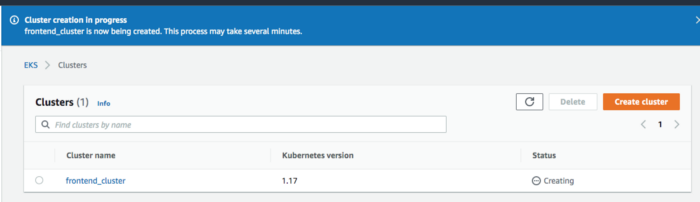
**eks role**

Go to the EKS console and configure the cluster as below. I used the kubernetes version 1.17 and you can check the Cluster service role.

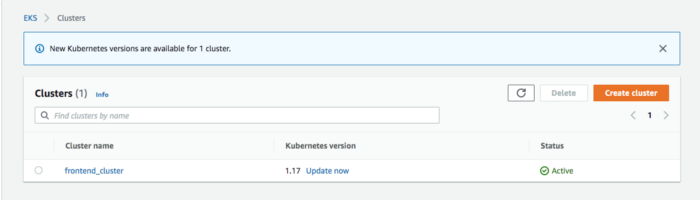


**Configure Cluster**

It takes some time for the cluster to get created and it should be in the active state once it is created.



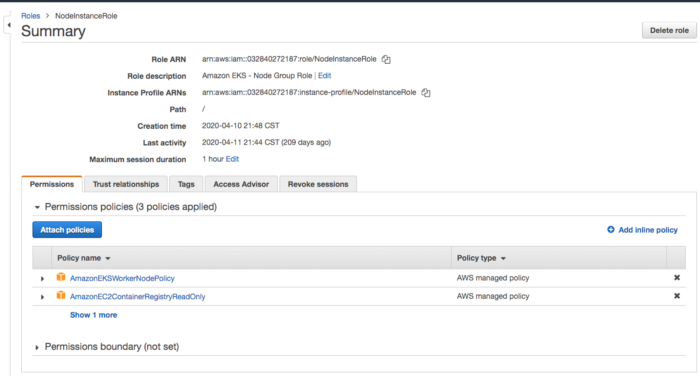
**Cluster Creating**



**Cluster is Active**

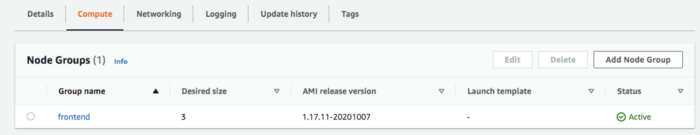
**Create Worker Nodes**

It’s time to create nodes and before you do that we have to create this role called NodeInstanceRole. [Follow this guide to create one.](https://docs.aws.amazon.com/eks/latest/userguide/worker_node_IAM_role.html)



**NodeInstanceRole**

[Follow this guide to create a node group after the role is created.](https://docs.aws.amazon.com/eks/latest/userguide/create-managed-node-group.html) It takes some time for the nodegroup to get created. You will see the static as Active once done.



**Node Group is active**

**Configure kubectl to use Cluster**

* We need to install kubectl on our machine, [follow this guide to install depending on your OS.](https://kubernetes.io/docs/tasks/tools/install-kubectl/)
* The next thing we need to do is to install an aws-iam-authenticator. [Follow this guide.](https://docs.aws.amazon.com/eks/latest/userguide/install-aws-iam-authenticator.html) We need this to authenticate the cluster and it uses the same user as AWS CLI is authenticated with.
* Use the AWS CLI **update-kubeconfig** command to create or update your kubeconfig for your cluster. Here region-code is **us-east-2** and cluster\_name is **frontend\_clutser**

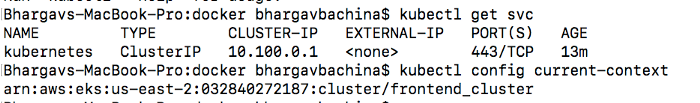
aws eks --region ***region-code*** update-kubeconfig --name ***cluster\_name***



**connected to cluster**

You can check with these commands.

// get the service  
kubectl get svc// get the current context  
kubectl config current-context



**Connected to AWS EKS**

**Deploy Kubernetes Objects On AWS EKS Cluster**

Now we have configured kubectl to use AWS EKS from our own machine. Let’s create deployment and service objects and use the image from the AWS ECR. Here is the manifest file which contains these objects.

**manifest.yaml**

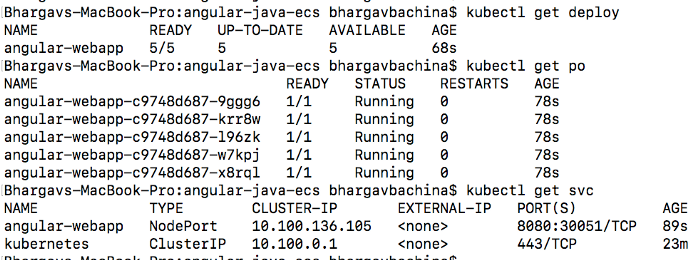
If you cloned the above example project and you are at the root folder just use this command to create objects kubectl create -f manifest.yml



**kubectl create -f manifest.yml**

You can use these following commands to verify all the objects are in the desired state.

// list the deployment  
kubectl get deploy// list the pods  
kubectl get po// list the service  
kubectl get svc



**all objects are deployed**

**Summary**

* Amazon Elastic Kubernetes Service (Amazon EKS) is a managed service that makes it easy for you to run Kubernetes on AWS without needing to stand up or maintain your own Kubernetes control plane.
* You need to create an AWS Account as a prerequisite.
* It’s not a best practice to use your root account to do any tasks instead you should create an IAM group that has permissions for administrator access and add a user to it and log in with that user.
* You should use this command aws configure with access key and secret key.
* Amazon EKS is a managed service that makes it easy for you to run Kubernetes on AWS.
* Amazon Elastic Container Registry (ECR) is a fully-managed [Docker](https://aws.amazon.com/docker/) container registry that makes it easy for developers to store, manage, and deploy Docker container images.
* Amazon ECR is integrated with [Amazon Elastic Container Service (ECS)](https://aws.amazon.com/ecs/), simplifying your development to production workflow.
* Amazon ECS works with any Docker registry such as Docker Hub, etc.
* You have to follow these steps to run apps on the Kubernetes cluster: we need to create an AWS EKS cluster with AWS console, SDK, or AWS CLI. Create a worker node group that registers with EKS Cluster, when your cluster is ready, you can configure **kubectl** to communicate with your cluster, Deploy and manage your applications on the cluster

**Conclusion**

We have deployed a simple Angular application with Java Backend on AWS EKS Cluster. In future posts, we will see how we can add a load balancer and how we can route requests to the Kubernetes Cluster.