

# Deploying React App With NodeJS Backend on AKS and storing the image on ACR

## Problem Statement –

Traditionally concept of using VM which are very slow & hard to manage. Even the VMs also not cost efficient. Docker came into the picture. Containers require less system resources than traditional or hardware virtual machine environments because they don't include operating system images. Applications running in containers can be deployed easily to multiple different operating systems and hardware platforms.

But handling the Multiple container was a very critical task. Even providing the high availability and having scalable containerised application was difficult to achieve .

## Introduction:-

Inorder to handle the multiple complex containers the Kubernetes comes into the picture. Kubernetes(also known as K8s) handles the container efficiently by creating the cluster. Kubernetes helps in providing the high availability & low latency of the application to the end user. Kubernetes extends how we scale containerized applications so that we can enjoy all the benefits of a truly immutable infrastructure. The general rule of thumb for K8S: if your app fits in a container, Kubernetes will deploy it.

## Kubernetes basic terms and definitions:

To begin understanding how to use K8S, we must understand the objects in the API. Basic K8S objects and several higher-level abstractions are known as **controllers**. These are the building block of your application lifecycle.

Basic objects include:

- **Pod.** A group of one or more containers.
- **Service.** An abstraction that defines a logical set of pods as well as the policy for accessing them.
- **Volume.** An abstraction that lets us persist data. (This is necessary because containers are ephemeral—meaning data is deleted when the container is deleted.)
- **Namespace.** A segment of the cluster dedicated to a certain purpose, for example a certain project or team of devs.

Controllers, or higher-level abstractions, include:

- **ReplicaSet (RS).** Ensures the [desired amount of pod](#) is what's running.
- **Deployment.** Offers declarative updates for pods an RS.
- **StatefulSet.** A workload API object that manages stateful applications, such as databases.
- **DaemonSet.** Ensures that all or some worker nodes run a copy of a pod. This is useful for daemon applications like [Fluentd](#).

- **Job.** Creates one or more pods, runs a certain task(s) to completion, then deletes the pod(s).

## Kubernetes architecture and components

A K8S cluster is made of a master node, which exposes the API, schedules deployments, and generally manages the cluster. Multiple worker nodes can be responsible for container runtime, like Docker or rkt, along with an agent that communicates with the master.

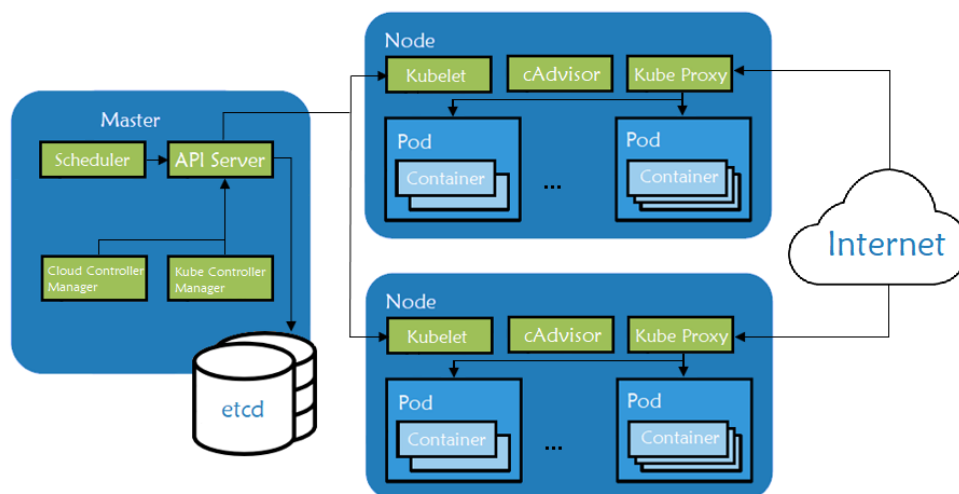
### Master components

These master components comprise a master node:

- **Kube-apiserver.** Exposes the API.
- **Etcd.** Key value stores all cluster data. (Can be run on the same server as a master node or on a dedicated cluster.)
- **Kube-scheduler.** Schedules new pods on worker nodes.
- **Kube-controller-manager.** Runs the controllers.
- **Cloud-controller-manager.** Talks to cloud providers.

### Node components

- **Kubelet.** Agent that ensures containers in a pod are running.
- **Kube-proxy.** Keeps network rules and perform forwarding.
- **Container runtime.** Runs containers.



Kubernetes Architecture

In order to create a K8s cluster on Azure Microsoft created a solution named as **Azure Kubernetes Service(AKS)** . AKS is Microsoft Azure's managed Kubernetes solution that lets you run and manage containerized applications in the cloud. Since this is a managed Kubernetes service, Microsoft takes care of a lot of things for us such as security, maintenance, scalability, and monitoring. This makes us quickly deploy our applications into the Kubernetes cluster without worrying about the underlying details of building it.

### **Solution:-**

Some steps and pre-requisite needs to be followed:

- **Cloning from github**
- **Install Azure CLI and Configure**
- **Dockerize the Project**
- **Pushing Docker Image To Container Registry**
- **Creating AKS Cluster**
- **Configure Kubectl With AKS Cluster**
- **Deploy Kubernetes Objects On Azure AKS Cluster**
- **Access the WebApp from the browser**



### **Cloning from github:**

**Github repository:** <https://github.com/imabhayvarshney/react-nodejs-aks.git>

```
$ git clone https://github.com/imabhayvarshney/react-nodejs-aks.git
Cloning into 'react-nodejs-aks'...
warning: You appear to have cloned an empty repository.
```

Name	Date modified	Type
api	23-11-2020 12:04 AM	File folder
my-app	23-11-2020 12:04 AM	File folder
react-nodejs-aks	11-01-2023 07:39 PM	File folder
Dockerfile	23-11-2020 12:04 AM	File
manifest.yml	23-11-2020 12:04 AM	Yaml Source File


## Install Azure CLI and Configure

Once you have the Azure Account you can install Azure CLI. You can go to the below documentation and install Azure CLI based on your operation system. You can configure Azure CLI with your subscription.

- Install Azure CLI
- Login into your account

```
PS C:\Users\Abhay_Varshney> az login
az : WARNING: A web browser has been opened at
https://login.microsoftonline.com/organizations/oauth2/v2.0/authorize. Please continue the login in the web
browser. If no web browser is available or if the web browser fails to open, use device code flow with `az
login --use-device-code`.
At line:1 char:1
+ az login
+ ~~~~~
+ CategoryInfo          : NotSpecified: (WARNING: A web ...e-device-code`.:String) [], RemoteException
+ FullyQualifiedErrorId : NativeCommandError
```

Microsoft Azure




**Sign in**

Email, phone, or Skype

---

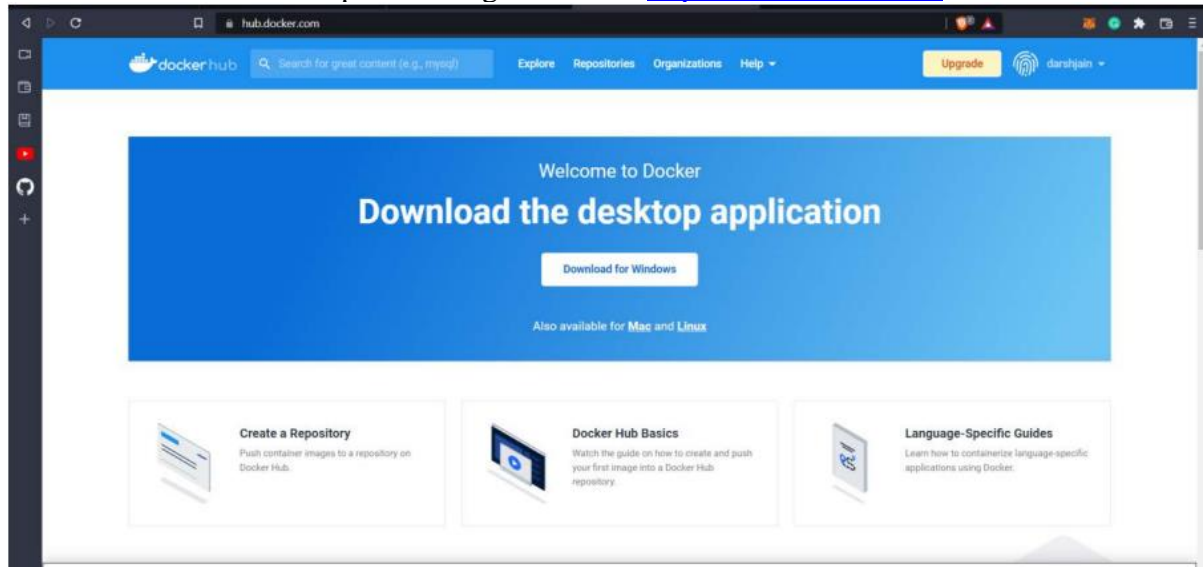
[No account? Create one!](#)  
[Can't access your account?](#)

Back
Next

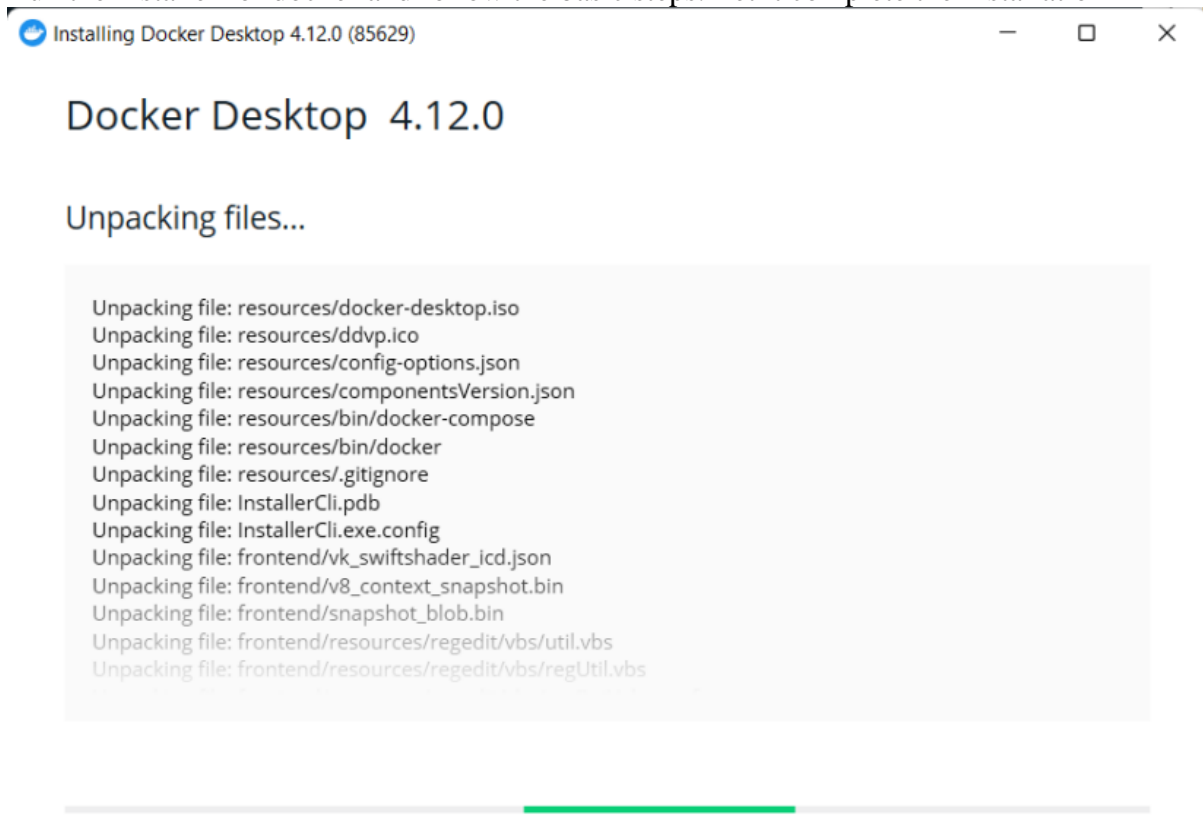
 Sign-in options

## Dockerize the Project

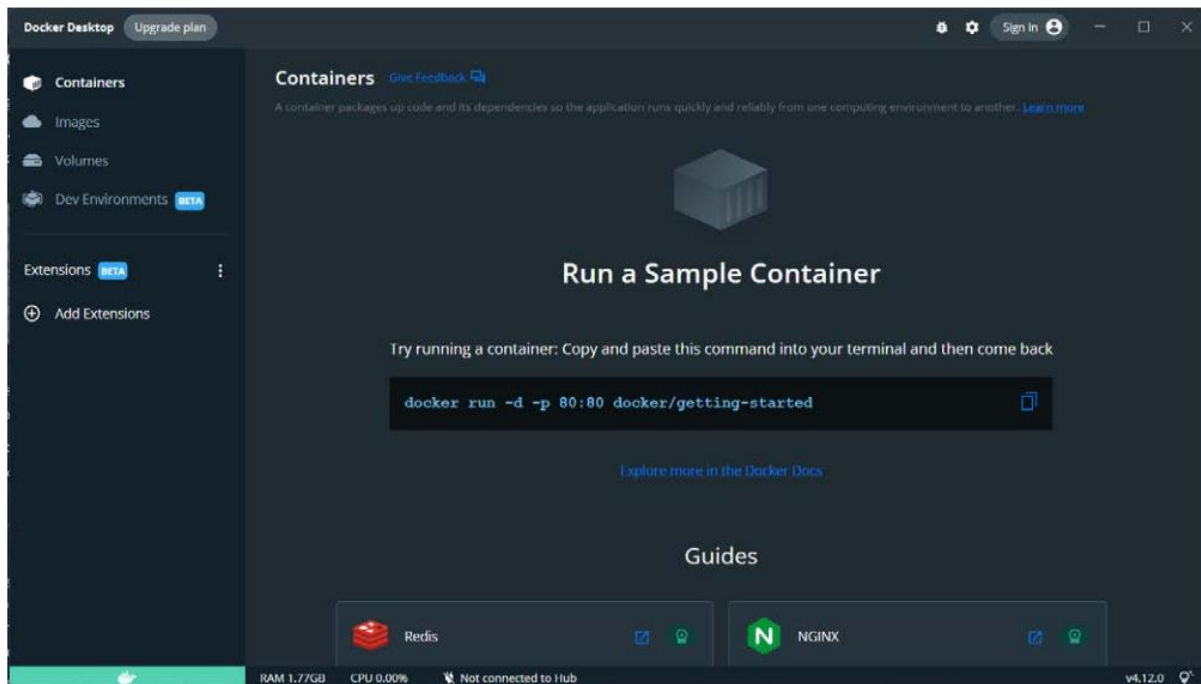
Download Docker Desktop from the given link: - <https://hub.docker.com>



Run the installer for docker and follow the basic steps. Let it complete the installation



Start the Docker Desktop service by launching the application. You should receive the following screen upon successful installation and is ready for work.



Azure AKS is a managed service that makes it easy for you to run Kubernetes on Azure. The first thing you need to do is to dockerize your project which we cloned from github. Here is the Dockerfile and it is using multi-stage builds to reduce the image size and surface attacks.

Dockerfile:

```
FROM node:10 AS ui-build
WORKDIR /usr/src/app
COPY my-app/ ./my-app/
RUN cd my-app && npm install && npm run build
FROM node:10 AS server-build
WORKDIR /root/
COPY --from=ui-build /usr/src/app/my-app/build ./my-app/build
COPY api/package*.json ./api/
RUN cd api && npm install
COPY api/server.js ./api/
EXPOSE 3080
CMD ["node", "./api/server.js"]
```

Here are the commands to build the image and run it on the Docker engine on your local machine. If you are new to Docker and check this detailed post on this topic.

## Dockerizing React App With NodeJS Backend

```
// build the image
docker build -t react-node-image .

// running on Image
docker run -it -p 3080:3080 --name react-node-ui react-node-image

// list the image you just built
docker images

// list the container
docker ps
```

## Pushing Docker Image To Container Registry

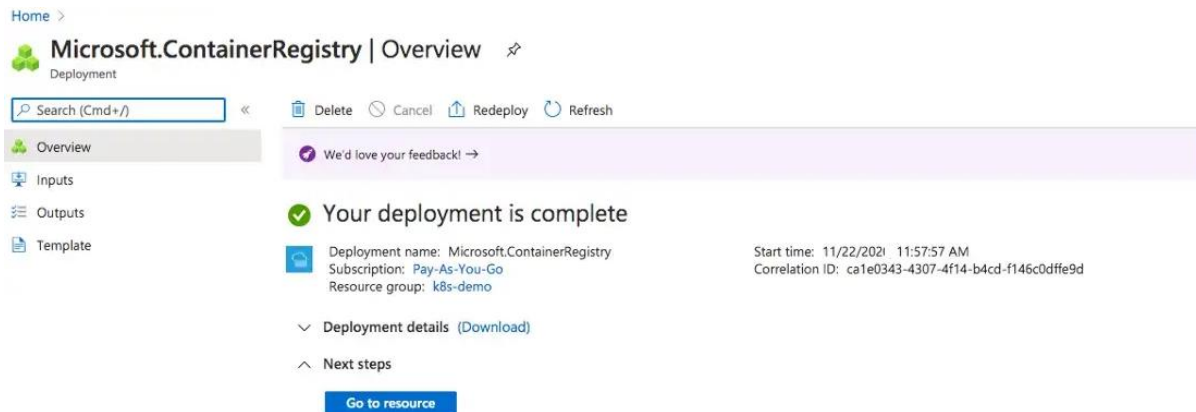
Azure container registry builds, store, secure, scan, replicate, and manage container images and artifacts with a fully managed, geo-replicated instance of OCI distribution. Connect across environments, including Azure Kubernetes Service and Azure Red Hat OpenShift, and across Azure services like App Service, Machine Learning, and Batch.

Azure AKS works with any Docker registry such as Docker Hub, etc. But, here, we see how we can use the Azure container registry to store our Docker images. Once you set up the Azure portal account and creates a resource group as above you can create a container registry as below.

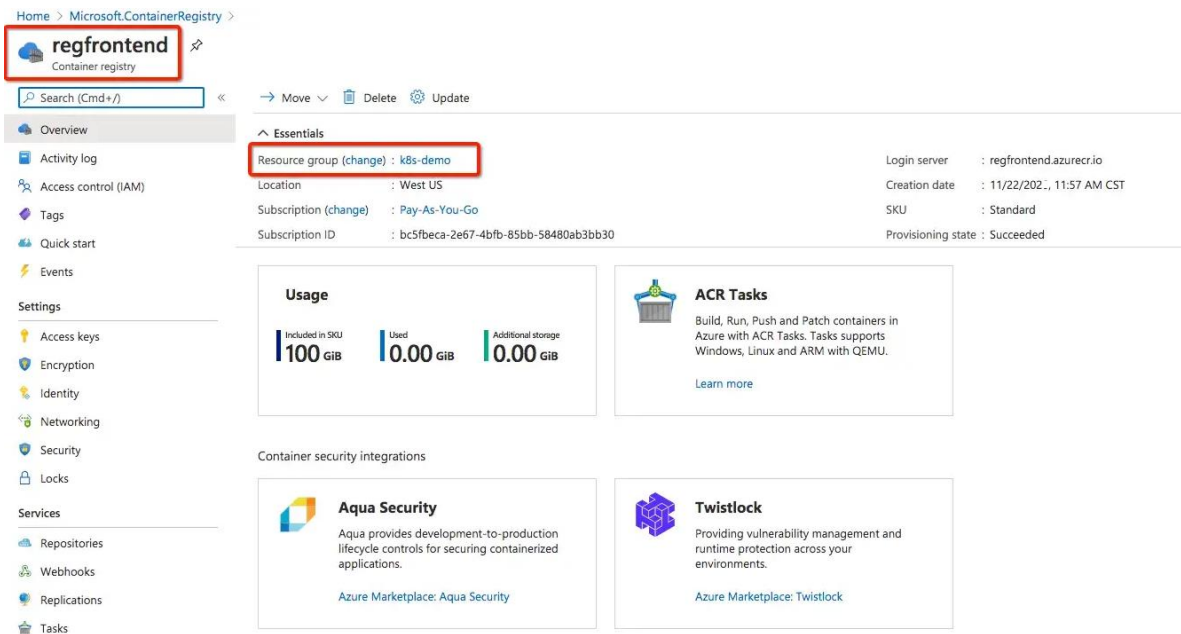
The screenshot shows the Azure portal interface for creating a container registry. The top navigation bar includes the Microsoft Azure logo and a search bar. Below the navigation, the page title is 'Create container registry'. The 'Basics' tab is selected, with other tabs for 'Networking', 'Encryption', 'Tags', and 'Review + create'. A brief description of Azure Container Registry is provided. The form is divided into 'Project details' and 'Instance details' sections. The 'Project details' section includes 'Subscription' (Pay-As-You-Go) and 'Resource group' (k8s-demo). The 'Instance details' section includes 'Registry name' (regfrontend), 'Location' (West US), and 'SKU' (Standard).

Section	Field	Value
Project details	Subscription *	Pay-As-You-Go
	Resource group *	k8s-demo
Instance details	Registry name *	regfrontend
	Location *	West US
	SKU * ⓘ	Standard

Once you review and create you can see the following screen.



You can see the main container registry page below.



You can do the same things with the Azure CLI with the following commands. Make sure you log in to your Azure Account with CLI with this command `az login` before running the below commands.



```
// create a resource group
az group create --name k8s-demo --location westus
```

```
// create a container registry
az acr create --resource-group k8s-demo \
--name regfrontend --sku Basic
```

It's time to build and push the Docker image with the following command. After cloning the above project and go to the root folder where Dockerfile resides and run this command.

```
az acr build --image aksdemo/react-nodejs:v1 \
--registry regfrontend \
--file Dockerfile .
```

You will see the output below

```
6ad22fbe53ce: Pushed
v1: digest: sha256:eb187a5e50607148e4b8c82703bc147807a1c19a54e44e9677995833a93f1add size: 3053
2020/11/22 18:07:20 Successfully pushed image: regfrontend.azurecr.io/aksdemo/react-nodejs:v1
2020/11/22 18:07:20 Step ID: build marked as successful (elapsed time in seconds: 160.981132)
2020/11/22 18:07:20 Populating digests for step ID: build...
2020/11/22 18:07:21 Successfully populated digests for step ID: build
2020/11/22 18:07:21 Step ID: push marked as successful (elapsed time in seconds: 58.736700)
2020/11/22 18:07:21 The following dependencies were found:
2020/11/22 18:07:21
- image:
  registry: regfrontend.azurecr.io
  repository: aksdemo/react-nodejs
  tag: v1
  digest: sha256:eb187a5e50607148e4b8c82703bc147807a1c19a54e44e9677995833a93f1add
runtime-dependency:
  registry: registry.hub.docker.com
  repository: library/node
  tag: "10"
  digest: sha256:14fa22a8989cd64ce811db9d47e3ed2910e0f2d95323240e23bc928201bbf313
  git: {}

Run ID: cf1 was successful after 3m47s
```

You can see all the details in the portal as well.

The screenshot displays the Microsoft Azure portal interface for a Container Registry. The breadcrumb navigation shows 'Home > Microsoft.ContainerRegistry > regfrontend > aksdemo/react-nodejs'. The main content area is titled 'regfrontend | Repositories' and includes a search bar, a refresh button, and a list of repositories. The 'aksdemo/react-nodejs' repository is selected and highlighted with a red box. The repository details panel on the right shows the repository name, a refresh button, a delete button, and essential information: 'Repository : aksdemo/react-nodejs', 'Tag count : 1', and 'Last updated date : 11/22/2020 , 12:07 PM CST'. Below this, there is a search bar for tags and a list of tags, with 'v1' visible. On the left sidebar, the 'Services' section is expanded, and 'Repositories' is highlighted with a red box.

**aksdemo/react-nodejs:v1**  
sha256:eb187a5e50607148e4b8c82703bc147807a1c19a54e4e9677995833a93f1add

Essentials

Repository	: aksdemo/react-nodejs	Digest	: sha256:eb187a5e50607148e4b8c82703bc147807a1c19a54e4e9677995833a93f1add
Tag	: v1	Manifest creation date	: 11/22/2020, 12:07 PM CST
Tag creation date	: 11/22/2020, 12:07 PM CST	Platform	: linux / amd64
Tag last updated date	: 11/22/2020, 12:07 PM CST	Run ID	: cf1

Docker pull command: `docker pull regfrontend.azurecr.io/aksdemo/react-nodejs:v1`

Manifest

```

{
  "schemaVersion": 2,
  "mediaType": "application/vnd.docker.distribution.manifest.v2+json",
  "config": {
    "mediaType": "application/vnd.docker.container.image.v1+json",
    "size": 9182,
    "digest": "sha256:c0f4199e10b1b447ce25514f27e8e12624402b997f99bb01b1fde919404466"
  },
  "layers": [
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 45377037,
      "digest": "sha256:7919f5b7d60254caf73c0d097b8ccfb72e0b6472957eace4dd5b378c5ca7cc1"
    },
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 19792132,
      "digest": "sha256:0e107167d0c5392ce7b34cb6af6bcf61c99f76f9e632d6c32152666558ab50"
    },
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 4340590,
      "digest": "sha256:66a456bba43b99e4c17dd5da957e3be72c43ae5291b055ce92bd2e49153259"
    },
    {
      "mediaType": "application/vnd.docker.image.rootfs.diff.tar.gzip",
      "size": 50110915,
      "digest": "sha256:5435318a0426be894e9142e90cb801d927e23cb27e109431e292d8fbc926"
    }
  ]
}

```

If you want to pull this repository you need to use this command.

```
docker pull regfrontend.azurecr.io/aksdemo/react-nodejs:v1
```

## Creating AKS Cluster

First, you need a resource group for all your resources. Let's create a resource with the following command.

```
az group create --name k8s-demo --location westus
```

Let's create a cluster with the following command. Notice that we are using the same resource group that we created above. You can see the JSON formatted result after a few minutes.

```
az aks create --resource-group k8s-demo --name frontend-cluster --node-count 3 --enable-addons monitoring --generate-ssh-keys
```

You can see the following cluster in the console.

The screenshot displays the Azure portal interface for a Kubernetes service. The main heading is 'frontend-cluster' with a sub-heading 'Kubernetes service'. The left sidebar contains navigation options like Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Namespaces (preview), Workloads (preview), Services and ingresses (preview), Storage (preview), Configuration (preview), Settings, Node pools, Configuration, Scale, Networking, Dev Spaces, and Deployment center (preview).

The main content area is divided into several sections:

- Essentials:**
  - Resource group (change): k8s-demo
  - Status: Succeeded
  - Location: West US
  - Subscription (change): Pay-As-You-Go
  - Subscription ID: bc5fbeca-2e67-4bfb-85bb-58480ab3bb30
  - Tags (change): Click here to add tags
- Properties:**
  - Kubernetes services:**
    - Kubernetes version: 1.17.13
    - Azure AD integration: Not enabled
  - Node pools:**
    - Node pools: 1 node pool
    - Kubernetes versions: 1.17.13
    - Node sizes: Standard\_DS2\_v2
    - Virtual node pools: Not enabled
- Networking:**
  - API server address: frontend-c-k8s-demo-bc5fbeca-d0ca98f1.hcp.westus.azmk8s.io
  - Network type (plugin): Kubenet
  - Private cluster: Not enabled
  - Pod CIDR: 10.244.0.0/16
  - Service CIDR: 10.0.0.0/16
  - DNS service IP: 10.0.0.10
  - Docker bridge CIDR: 172.17.0.1/16
  - HTTP application routing: Not enabled
- Integrations:**
  - Container insights: Enabled
  - Workspace resource ID: defaultworkspace-bc5fbeca-2e67-4bfb-85bb-58480

## Configure Kubectl With AKS Cluster

Kubectl is the command-line utility for the Kubernetes. You need to install kubectl before you configure it. Run the first command only if you don't have kubectl on your local machine.

```
// install CLI
az aks install-cli

// connect to your cluster
az aks get-credentials --resource-group k8s-demo --name frontend-cluster

// get all the contexts
kubectl config get-contexts

// verify the current context
kubectl config current-context

// get the node
kubectl get nodes
```

```
-MacBook-Pro:next $ az aks get-credentials --resource-group k8s-demo --name frontend-cluster
Merged "frontend-cluster" as current context in /Users/bhargavbachina/.kube/config
-MacBook-Pro:next $ kubectl config current-context
frontend-cluster
-MacBook-Pro:next $ kubectl get nodes
NAME                                STATUS    ROLES    AGE    VERSION
aks-nodepool1-95737313-vmss000000  Ready    agent    5m59s  v1.17.13
aks-nodepool1-95737313-vmss000001  Ready    agent    5m52s  v1.17.13
aks-nodepool1-95737313-vmss000002  Ready    agent    5m53s  v1.17.13
```

## Deploy Kubernetes Objects on Azure AKS Cluster

Now we have configured kubectl to use Azure AKS from our own machine. You need to integrate the container registry with the AKS. Let's attach the container registry with the cluster with the following command.

```
az aks update -n frontend-cluster -g k8s-demo --attach-acr regfrontend
```

Let's create deployment and service objects and use the image from the Azure container registry. Here is the manifest file which contains these objects.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: react-webapp
  name: react-webapp
spec:
  replicas: 5
  selector:
    matchLabels:
      app: react-webapp
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: react-webapp
    spec:
      containers:
        - image: regfrontend.azurecr.io/aksdemo/react-nodejs:v1
          name: webapp
          imagePullPolicy: Always
          resources: {}
          ports:
            - containerPort: 3080
status: {}

---

apiVersion: v1
kind: Service
metadata:
  name: react-webapp
  labels:
```

```
  run: react-webapp
spec:
  ports:
  - port: 3080
    protocol: TCP
  selector:
    app: react-webapp
  type: LoadBalancer
```

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
```

```
-MacBook-Pro:react-nodejs-aks ~ % kubectl create -f manifest.yml
deployment.apps/react-webapp created
service/react-webapp created
```

#### **K8s objects created**

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

```
// list the deployment
kubectl get deploy

// list the pods
kubectl get pod

// list the service
kubectl get svc
```

We can see 5 pods running since we have defined 5 replicas for the deployment.

```

-MacBook-Pro:react-nodejs-aks $ kubectl get deploy
NAME          READY   UP-TO-DATE   AVAILABLE   AGE
react-webapp  5/5     5             5           37s

-MacBook-Pro:react-nodejs-aks $ kubectl get po
NAME                                READY   STATUS    RESTARTS   AGE
react-webapp-b7ff855ff-92ttv        1/1     Running   0           43s
react-webapp-b7ff855ff-mm6h6        1/1     Running   0           43s
react-webapp-b7ff855ff-q2mdk        1/1     Running   0           43s
react-webapp-b7ff855ff-qcmjtf       1/1     Running   0           43s
react-webapp-b7ff855ff-t98wx        1/1     Running   0           43s

-MacBook-Pro:react-nodejs-aks $ kubectl get svc
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes   ClusterIP     10.0.0.1        <none>           443/TCP          116m
react-webapp  LoadBalancer 10.0.157.101    13.86.186.222   3080:32210/TCP  50s

```

## Access the WebApp from the browser

We have created a service with the LoadBalancer type. You can get the external IP from the service and access the entire from the browser.

```

-MacBook-Pro:react-nodejs-aks $ kubectl get svc
NAME          TYPE          CLUSTER-IP      EXTERNAL-IP      PORT(S)          AGE
kubernetes   ClusterIP     10.0.0.1        <none>           443/TCP          116m
react-webapp  LoadBalancer 10.0.157.101    13.86.186.222   3080:32210/TCP  50s

```

service

You can access the webapp with the following URL

```
http://13.86.186.222:3080
```

The screenshot shows a web browser window with the address bar containing `13.86.186.222:3080`. The page displays a React application titled "React With NodeJS". On the left, there is a "Create User" form with input fields for "First Name" (containing "sfsdf"), "Last Name" (containing "sfsdf"), and "Email" (containing "sfsdfsd"), along with a "Create" button. On the right, a pink notification box shows "Users Created" with a large red "1" and a "Get all Users" button. Below the form, a table titled "Users" contains one row of data:

User Id	Firstname	Lastname	Email
1	sfsdf	sfsdf	sfsdfsd

