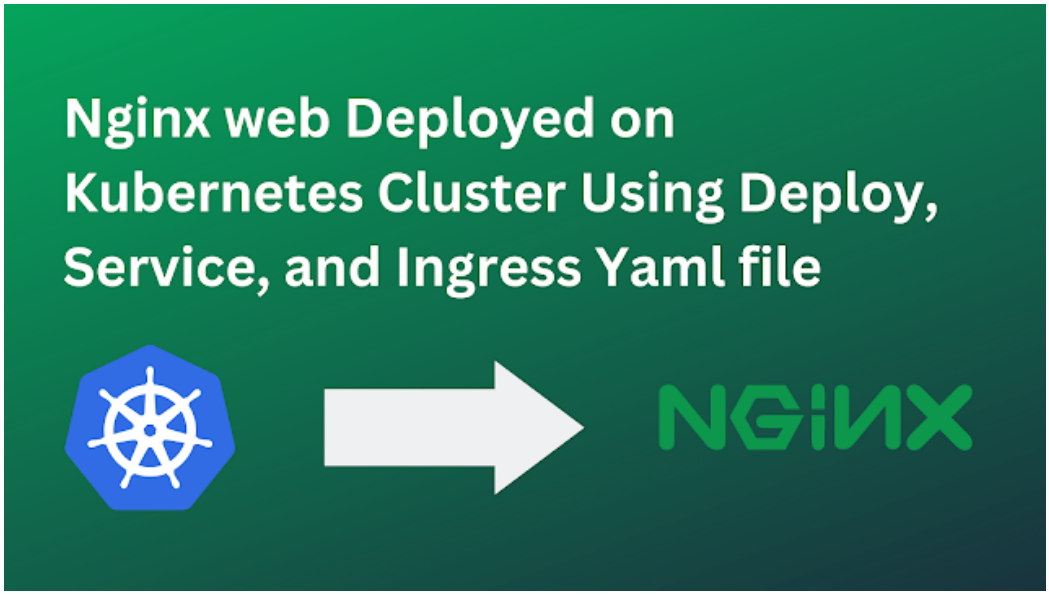
Nginx Web Deployed on Kubernetes Cluster Using Deploy, Service, and Ingress Yaml Files

April 07, 2023

In this blog, we are going to explore how to deploy nginx to Kubernetes using a Deployment file, Service file, and Ingress file. These files are essential components of any Kubernetes application deployment, and they work together to ensure that your application is available and accessible to your users.



We will start by creating a Deployment file that specifies the desired state of our nginx deployment, including the container image to use, the number of replicas to create, and any necessary environment variables. Next, we will create a Service file that exposes our nginx deployment to other pods within our Kubernetes cluster, allowing them to communicate with each other. Finally, we will create an

request.

Once we have created these files, we will use kubectl to deploy our nginx application to our Kubernetes cluster. We will also use ngrok to create a secure tunnel to our Kubernetes cluster, allowing us to preview our application in a live environment.

By the end of this blog, you will have a solid understanding of how to deploy and manage applications on Kubernetes using these essential components, as well as how to preview your application in a live environment using ngrok.

Before you begin

This tutorial assumes that you are using minikube and docker to run a local Kubernetes cluster. Visit Install tools to learn how to install minikube.

You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. It is recommended to run this tutorial on a cluster with at least two nodes that are not acting as control plane hosts. If you do not already have a cluster, you can create one by using minikube or you can use one of these Kubernetes playgrounds:

minikube start

minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.

All you need is Docker (or similarly compatible) container or a Virtual Machine environment, and Kubernetes is a single command away: minikube start

2 CPUs or more

2GB of free memory

20GB of free disk space

Internet connection

Container or virtual machine manager, such as: Docker, QEMU, Hyperkit, Hyper-V, KVM, Parallels, Podman, VirtualBox, or VMware Fusion/Workstation

Command for minikube

New-Item -Path 'c:\' -Name 'minikube' -ItemType Directory -Force

Invoke-WebRequest -OutFile 'c:\minikube\minikube.exe' -Uri 'https://github.com/kubernetes/minikube/releases/latest/download/minikube-windows-amd64.exe' -UseBasicParsing

Start your cluster

minikube start

Interact with your cluster

First kubectl install, then you can now use it to access your shiny new cluster:

kubectl get pods

Alternatively, minikube can download the appropriate version of kubectl and you should be able to use it like this:

Deploy applications

Deployment YAML: A Deployment YAML file is used to define a Kubernetes Deployment. Deployments are used to manage the

YAML file, you can specify the container image, the number of replicas to create, and various other configuration options. Here's an example of what a simple Deployment YAML file might look like:

```
ubuntu@ip-172-31-91-225:~/rahul$ cat deploy.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

Now run this command

kubectl apply -f deploy.yaml

Created container for Deployment file

Service YAML:

A Service YAML file is used to define a Kubernetes Service. Services provide a stable IP address and DNS name for accessing a set of Pods, even as the Pods come and go. In a Service YAML file, you can specify the type of Service (ClusterIP, NodePort, or LoadBalancer), the port(s) to expose, and the selector to use for determining which Pods to route traffic to. Here's an example of what a simple Service YAML file might look like:

```
  app: nginx
ports:
  - port: 80
    targetPort: 80
    nodePort: 30007
```

Now run this command

kubectl apply -f service.yaml

Ingress YAML:

An Ingress YAML file is used to define a Kubernetes Ingress. Ingresses provide a way to route external traffic to Services within your cluster. In an Ingress YAML file, you can specify the rules for how traffic should be routed based on the host or path of the incoming request. Here's an example of what a simple Ingress YAML file might look like:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: nginx
spec:
  rules:
  - host: "digital.com"
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: my-service
            port:
              number: 80
ubuntu@ip-172-31-91-225:~/rahul$
```

Now run this command

kubectl apply -f ingress.yaml

Now need to check IP on same Instance check by following command.

curl -L "ip of service"

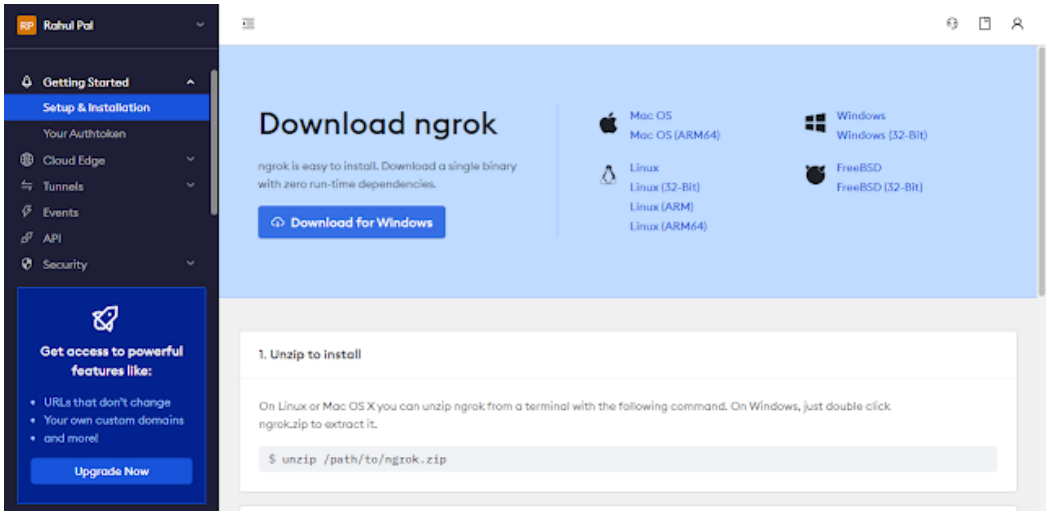
dex, follow, max-image-preview:large, max-snippet:-1, max-video-preview:-1' /> <style>/*Start: New Critical Part*/@font-face{font-family:"Inter";font-style:normal;font-weight:200;font-display:swap;src:url("/wp-content/themes/digital-theme/assets/fonts/Inter.woff2") format("woff2"),url(https://fonts.gstatic.com/s/inter/v3/UcC03FwrK3iLTeHuS_fvOtMwCp50KnMw2boKoduKmMEVuDyfMZs.woff) format("woff"),url(https://fonts.gstatic.com/s/inter/v3/UcC03FwrK3iLTeHuS_fvOtMwCp50KnMw2boKoduKmMEVuDyfMZg.ttf) format("truetype");unicode-range:U+0000-00FF,U+0131,U+0152-0153,U+02BB-02BC,U+02C6,U+02DA,U+02DC,U+2000-206F,U+2074,U+20AC,U+2122,U+2191,U+2193,U+2212,U+2215,U+FEFF,U+FFFD}@font-face{font-family:"Inter";font-style:normal;font-weight:300;font-display:swap;src:url("/wp-content/themes/digital-theme/assets/fonts/Inter.woff2") format("woff2"),url(https://fonts.gstatic.com/s/inter/v3/UcC03FwrK3iLTeHuS_fvOtMwCp50KnMw2boKoduKmMEVuOKfMZs.woff) format("woff"),url(https://fonts.gstatic.com/s/inter/v3/UcC03FwrK3iLTeHuS_fvOtMwCp50KnMw2boKoduKmMEVuOKfMZg.ttf) format("truetype");unicode-range:U+0000-00FF,U+0131,U+0152-0153,U+02BB-02BC,U+02C6,U+02DA,U+02DC,U+2000-206F,U+2074,U+20AC,U+2122,U+2191,U+2193,U+2212,U+2215,U+FEFF,U+FFFD}@font-face{font-family:"Inter";font-style:normal;font-weight:400;font-display:swap;src:url("/wp-content/themes/digital-theme/assets/fonts/Inter.woff2") format("woff2"),url(https://fonts.gstatic.com/s/inter/v3/UcC03FwrK3iLTeHuS_fvOtMwCp50KnMw2boKoduKmMEVuLyfMZs.woff) format("woff"),url(https://fonts.gstatic.com/s/inter/v3/UcC03FwrK3iLTeHuS_fvOtMwCp50KnMw2boKoduKmMEVuLyfMZg.ttf) format("truetype");unicode-range:U+0000-00FF,U+0131,U+0152-0153,U+02BB-02BC,U+02C6,U+02DA,U+02DC,U+2000-206F,U+2074,U+20AC,U+2122,U+2191,U+2193,U+2212,U+2215,U+FEFF,U+FFFD}@font-face{font-family:"Inter";font-style:normal;font-weight:500;f

Now everything is fine there. So its time to show our running application to Developer, which is very important

Also Read: Guide: Deploying WordPress using IAM Role, RDS, Docker, ECS, and pushing it to ECR

Setup and create on ngrok

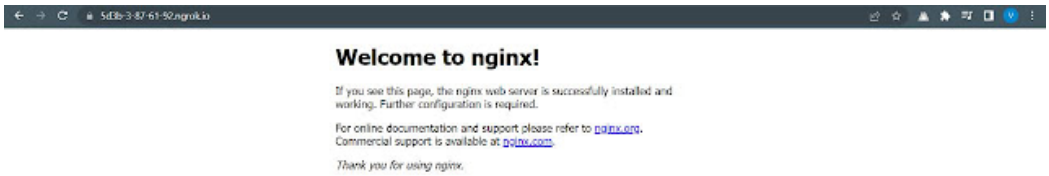
Now setup setup ngrok by instruction given by official website.



Collect Ip of our cluster

minikube service my-service –url

ngrok http http://192.168.49.2:30007



Boom! Our project is now live on Kubernetes and accessible to our users.

In conclusion,

Deploying nginx to Kubernetes using a Deployment file, Service file, and Ingress file is a straightforward process that can be accomplished with just a few commands using kubectl. These files work together to ensure that your application is available and accessible to your users, while also providing the flexibility to scale and manage your application as your needs change.

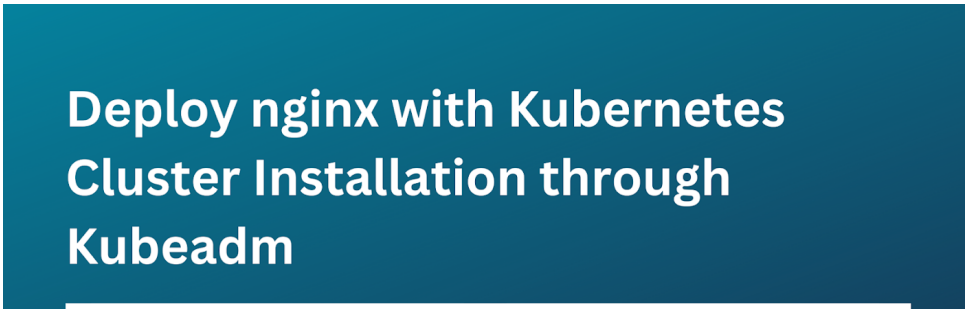
Additionally, by using ngrok to create a secure tunnel to your Kubernetes cluster, you can preview your application in a live environment, making it easier to test and troubleshoot any issues that may arise.

Overall, Kubernetes provides a powerful platform for deploying and managing your applications, and by following the steps outlined in

has to offer.

DEVOPS

Enter comment



Deploy nginx with Kubernetes Cluster Installation through Kubeadm

First we need to create 2 EC2 instances i.e t2 medium for master node and t2 micro for worker node. And allow ports for their connection. Run all below command on both master and worker nodes sudo apt update -y sudo apt install docker.io -y sudo systemctl start docker sud ...

Guide: Deploying WordPress using IAM Role, RDS, Docker, ECS, and pushing it to ECR

Before we get started, it's important to understand the role of each component in this deployment: IAM role: Allows EC2 instances to securely access AWS services such as RDS without needing to store AWS credentials on the instance itself. Docker: A containerization platform th ...