

# Create a Kubernetes Cluster And Deploy Jenkins, Using Elastic Kubernetes Service(EKS) :

## EC2 Control Node Instance Type:

t2.micro

AMI: Ubuntu.

## Security Group Settings:

- Ports to Open:
  - 22 for SSH
  - 80 for HTTP
  - 443 for HTTPS

The screenshot shows the 'Launch an instance' page in the AWS Management Console. The page is divided into several sections:

- Header:** A navigation bar at the top contains icons for various AWS services: EC2, Lambda, CloudShell, CloudWatch, IAM, Billing, DynamoDB, CloudFront, Elastic Beanstalk, S3, CodeCommit, and Elastic Container Registry.
- Main Title:** 'Launch an instance' with an 'Info' link.
- Introduction:** A paragraph stating 'Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.'
- Name and tags:** A section with a text input field containing 'K8S Control Node' and a link to 'Add additional tags'.
- Application and OS Images (Amazon Machine Image):** A section with a search bar and a 'Quick Start' section. The 'Quick Start' section displays a grid of AMI icons for Amazon Linux, macOS, Ubuntu (selected), Windows, Red Hat, and SUSE Linux. To the right is a 'Browse more AMIs' link.
- Amazon Machine Image (AMI) details:** A section showing the selected AMI: 'Ubuntu Server 22.04 LTS (HVM), SSD Volume Type' with its ID 'ami-0fc5d935ebf8bc3bc' and architecture '(64-bit (x86)) / (64-bit (Arm))'. It also shows 'Virtualization: hvm', 'ENA enabled: true', and 'Root device type: ebs'. A 'Free tier eligible' badge is present.
- Summary:** A sidebar on the right containing a 'Number of instances' input field set to '1', and links for 'Software Image (AMI)', 'Virtual server type (instance type)' (t2.micro), 'Firewall (security group)', 'New security group', and 'Storage (volumes)' (1 volume(s) - 8 GiB).
- Actions:** At the bottom right of the summary section are 'Cancel', 'Launch instance' (in an orange button), and 'Review commands' links.

Details
Security
Networking
Storage
Status checks
Monitoring
Tags

▼ Security details

IAM Role
-

Owner ID
771593319939

Launch time
Mon Oct 23 2023 19:05:57 GMT+0300 (Israel Daylight Time)

Security groups
sg-0bc3a4018f8784125 (K8S-Master-SecGroup)

▼ Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security g
-	sgr-0b731e89980ca4a20	443	TCP	0.0.0.0/0	<a href="#">K8S-Maste</a>
-	sgr-0db77f31d9b497beb	80	TCP	0.0.0.0/0	<a href="#">K8S-Maste</a>
-	sgr-01a3197467687bd43	22	TCP	87.68.218.115/32	<a href="#">K8S-Maste</a>

▼ Outbound rules

Name	Security group rule ID	Port range	Protocol	Destination	Security g
-	sgr-08dbc003d6973958	All	All	0.0.0.0/0	<a href="#">K8S-Maste</a>

## SSH into EC2 Instance

```
ssh -i "your-key.pem" ubuntu@<EC2_PUBLIC_IP>
```

## Change Host Name(Optional for comfort)

```
sudo hostnamectl set-hostname k8s-master
echo "127.0.0.1 k8s-master" | sudo tee -a /etc/hosts
```

(close the current SSH session and open a new one.  
The changes should take effect)

## Update Packages

```
sudo apt-get update
```

## Install AWS CLI

```
sudo apt install awscli -y
```

## Configure AWS CLI

Run the “aws configure” command(Here you'll be prompted to enter details such as):

- AWS Access Key ID: Your AWS Access Key.
- AWS Secret Access Key: Your AWS Secret Access Key.
- Default region name: The region where you want to create the EKS cluster.
- Default output format: You can leave this as ‘None’

## Install EKSCtl

```
curl --silent --location  
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname  
-s)_amd64.tar.gz" | tar xz -C /tmp  
sudo mv /tmp/eksctl /usr/local/bin
```

\*verify the installation

eksctl version

## Install Kubectl

```
sudo apt-get update  
sudo apt-get install -y apt-transport-https  
sudo curl -fsSL https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key add -  
echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a  
/etc/apt/sources.list.d/kubernetes.list  
sudo apt-get update  
sudo apt-get install -y kubectl
```

## Create an EKS Cluster

```
eksctl create cluster \  
--name "cluster-name" (You'll use this name to refer to the cluster) \  
--region "your-region" \  
--nodes 1 \  
--nodegroup-name "group-name" \  
--node-type "instance-type" (e.g t2/t3.medium) \  
--managed(simplifies node management.)
```

```
2023-10-23 18:18:57 [i] creating EKS cluster "devops-cluster" in "us-east-1" region with managed nodes  
2023-10-23 18:18:57 [i] will create 2 separate CloudFormation stacks for cluster itself and the initial managed nodegroup  
2023-10-23 18:18:57 [i] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-east-1 --cluster=devops-cluster'  
2023-10-23 18:18:57 [i] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "devops-cluster" in "us-east-1"  
2023-10-23 18:18:57 [i] CloudWatch logging will not be enabled for cluster "devops-cluster" in "us-east-1"  
2023-10-23 18:18:57 [i] you can enable it with 'eksctl utils update-cluster-logging --enable-types={SPECIFY-YOUR-LOG-TYPES-HERE (e.g. all)} --region=us-east-1 --cluster=devops-cluster'  
2023-10-23 18:18:57 [i]  
2 sequential tasks: { create cluster control plane "devops-cluster",  
  2 sequential sub-tasks: {  
    wait for control plane to become ready,  
    create managed nodegroup "devops-nodes",  
  }  
}  
2023-10-23 18:18:57 [i] building cluster stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:18:57 [i] deploying stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:19:27 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:19:57 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:20:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:21:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:22:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:23:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:24:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:25:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:26:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:27:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:28:58 [i] waiting for CloudFormation stack "eksctl-devops-cluster-cluster"  
2023-10-23 18:30:59 [i] building managed nodegroup stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:30:59 [i] deploying stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:30:59 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:31:29 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:32:25 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:33:15 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:34:17 [i] waiting for CloudFormation stack "eksctl-devops-cluster-nodegroup-devops-nodes"  
2023-10-23 18:34:17 [i] waiting for the control plane to become ready  
2023-10-23 18:34:17 [✓] EKS cluster "devops-cluster" in "us-east-1" region is ready
```

the output should be EKS cluster "cluster-name" in "region name" region is ready.

## Confirm the cluster creation

`eksctl get cluster --name 'cluster-name' --region "your region"`

```
ubuntu@k8s-master:~$ eksctl get cluster --name devops-cluster --region us-east-1
NAME          VERSION STATUS  CREATED              VPC          SUBNETS
devops-cluster 1.27    ACTIVE  2023-10-23T19:50:30Z vpc-025f7501f93fcbb6f subnet-2,subnet-0ac2d4118f68f1212
sg-09292cd9ec15b3fe6 EKS
```

## Verify that kubectl is properly communicating with your new cluster

`kubectl version`

```
ubuntu@k8s-master:~$ kubectl version
Client Version: v1.28.2
Kustomize Version: v5.0.4-0.20230601165947-6ce0bf390ce3
Server Version: v1.27.6-eks-f8587cb
```

## Create 'devops' namespace

`kubectl create namespace devops`

## Allocating An Elastic IP

Navigate to EC2 -> Elastic IP's->Allocate Elastic IP Address -> Allocate -> Choose the New IP, Actions, Associate elastic ip address -> Choose "Network Interface", The Instance Network Interface, His Private IP-> Associate

Elastic IP addresses (1/1)

Filter Elastic IP addresses

Actions

Allocate Elastic IP address

< 1 >

<input checked="" type="checkbox"/>	Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS record	Associated instance ID	Private IP addi
<input checked="" type="checkbox"/>	eksctl-devops-cluster-cluster/NATIP	44.205.211.83	Public IP	eipalloc-05caf0e5f7990fd22	-	-	192.168.4.209

## Allocate Elastic IP address Info

### Elastic IP address settings Info

Network Border Group Info

us-east-1

X

Public IPv4 address pool

☒ Amazon's pool of IPv4 addresses

☐ Public IPv4 address that you bring to your AWS account with BYOIP. (option disabled because no pools found) [Learn more](#)

☐ Customer-owned pool of IPv4 addresses created from your on-premises network for use with an Outpost. (option disabled because no customer owned pools found) [Learn more](#)

Global static IP addresses

AWS Global Accelerator can provide global static IP addresses that are announced worldwide using anycast from AWS edge locations. This can help improve the availability and latency for your user traffic by using the Amazon global network. [Learn more](#)

Create accelerator

Tags - optional

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

Add new tag

You can add up to 50 more tag

Cancel

Allocate

Elastic IP addresses (1/2)

Filter Elastic IP addresses

Actions

Allocate Elastic IP address

< 1 >

<input checked="" type="checkbox"/>	Name	Allocated IPv4 add...	Type	Allocation ID	Reverse DNS record	Private IP addi
<input checked="" type="checkbox"/>	-	174.129.2.204	Public IP	eipalloc-0d74ed994f1f8045a	-	-
<input type="checkbox"/>	eksctl-devops-cluster-cluster/NATIP	44.205.211.83	Public IP	eipalloc-05caf0e5f7990fd22	-	192.168.4.209

View details

Release Elastic IP addresses

Associate Elastic IP address

Disassociate Elastic IP address

Update reverse DNS

Enable transfers

Disable transfers

Accept transfers

## Associate Elastic IP address [Info](#)


Choose the instance or network interface to associate to this Elastic IP address (174.129.2.204)

### Elastic IP address: 174.129.2.204

#### Resource type


Choose the type of resource with which to associate the Elastic IP address.

- ☐ Instance
- ☒ Network interface

 If you associate an Elastic IP address with an instance that already has an Elastic IP address associated, the previously associated Elastic IP address will be disassociated, but the address will still be allocated to your account. [Learn more](#)

If no private IP address is specified, the Elastic IP address will be associated with the primary private IP address.

#### Network interface

 eni-054f6ee95e0e7dab6



#### Private IP address

The private IP address with which to associate the Elastic IP address.

 192.168.1.33



#### Reassociation

Specify whether the Elastic IP address can be reassociated with a different resource if it already associated with a resource.

- ☐ Allow this Elastic IP address to be reassociated

Cancel

Associate

## Create a Jenkins Dockerfile With necessary installations

```
Dockerfile x
home > ori > Desktop > Dockerfile
1
2
3 # Switch to root to install Docker and sudo
4 USER root
5
6 # Update the package index and install sudo
7 RUN apt-get update && \
8     apt-get install -y sudo && \
9     echo 'jenkins ALL=(ALL) NOPASSWD: ALL' >> /etc/sudoers
10
11 # Update the package index and install Docker
12 RUN apt-get update && \
13     apt-get install -y apt-transport-https ca-certificates curl software-properties-common && \
14     curl -fsSL https://download.docker.com/linux/ubuntu/gpg | apt-key add - && \
15     add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable" && \
16     apt-get update && \
17     apt-get install -y docker-ce
18
19 # ...
20 # Add Jenkins user to the docker group
21 RUN if getent group docker > /dev/null; then \
22     echo "The docker group already exists."; \
23     else \
24         groupadd docker; \
25     fi && \
26     usermod -aG docker jenkins
27 # ...
28
29 # Switch back to the Jenkins user
30 USER jenkins
31
```

ⓘ You have Docker installed on your system. Please install the recommended extensions.

**Build the Image:** Navigate to the directory where the Dockerfile is located, then build the Docker image:

**docker build -t <jenkins-image> .**

**docker tag <jenkins-image> <username>/<repo-name>:<tag>**

**docker push <username>/<repo-name>:<tag>**

```

ori@OriElias:~/Desktop$ docker build -t jenkins-image
ERROR: "docker buildx build" requires exactly 1 argument.
See 'docker buildx build --help'.

Usage:  docker buildx build [OPTIONS] PATH | URL | -

Start a build
ori@OriElias:~/Desktop$ docker build -t jenkins-image .
[+] Building 7.4s (7/7) FINISHED
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 131B
=> [internal] load metadata for docker.io/jenkins/jenkins:ls
=> [auth] jenkins/jenkins:pull token for registry-1.docker.io
=> CACHED [1/2] FROM docker.io/jenkins/jenkins:ls@sha256:80587de2dac2
=> [2/2] RUN apt-get update && apt-get install -y sudo
=> exporting to image
=> => exporting layers
=> => writing image sha256:1c13a55d0e309d18efde7331f88c4446aa4d7ffcde2
=> => naming to docker.io/library/jenkins-image
ori@OriElias:~/Desktop$ docker tag jenkins-image orielias/jenkins-image:latest
ori@OriElias:~/Desktop$ docker push orielias/jenkins-image:latest
The push refers to repository [docker.io/orielias/jenkins-image]
f0d828a9dc8f: Pushed
f20a0c613bdc: Mounted from jenkins/jenkins
599f5f8d0ab0: Mounted from jenkins/jenkins
4c8b96ce47c2: Mounted from jenkins/jenkins
44fc60778258: Mounted from jenkins/jenkins
5f7c5aca7eca: Mounted from jenkins/jenkins
c91efd33f121: Mounted from jenkins/jenkins
e68a7d6d6b63: Mounted from jenkins/jenkins
9cbc357f747e: Mounted from jenkins/jenkins
a9a7c0db4243: Mounted from jenkins/jenkins
965c6f166415: Mounted from jenkins/jenkins
78880e985c37: Mounted from jenkins/jenkins
2fa37f2ee66e: Mounted from jenkins/jenkins
latest: digest: sha256:55149de1332c5e6899969cceb059b8eb27bdf5604b0d526ff47b52d8d46f551f size: 3045
ori@OriElias:~/Desktop$

```



## Create a Jenkins persistent volume YAML File

change values : to your "node name"

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: local-pv
spec:
  capacity:
    storage: 10Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: local-storage
  local:
    path: /var/jenkins-data
  nodeAffinity:
    required:
      nodeSelectorTerms:
        - matchExpressions:
            - key: kubernetes.io/hostname
              operator: In
              values:
                - ip-192-168-0-18.ec2.internal
```

kubectl apply -f persistent-volume.yaml

## Create a Jenkins persistent volume claim YAML File

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: local-pvc
  namespace: devops
spec:
  storageClassName: local-storage
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 10Gi
```

kubectl apply -f persistent-volume-claim.yaml

## Create a Jenkins persistent local-storage YAML File

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: local-storage
provisioner: kubernetes.io/no-provisioner
volumeBindingMode: WaitForFirstConsumer
```

kubectl apply -f local-storage.yaml

## Expose the Jenkins Service

A NodePort service will open a specific port on all the cluster nodes, allowing external traffic to reach the service. (we specified 30000)

```

apiVersion: v1
kind: Service
metadata:
  name: jenkins
  namespace: devops
spec:
  type: NodePort
  ports:
    - port: 8080
      targetPort: 8080
      nodePort: 30000 # Optional
  selector:
    app: jenkins

```

## Apply the Service

kubectl apply -f jenkins-service.yaml

the output should be : “service/jenkins created”

## Create a Jenkins Deployment YAML File

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: jenkins
  namespace: devops
spec:
  replicas: 1
  selector:
    matchLabels:
      app: jenkins
  template:
    metadata:
      labels:
        app: jenkins
    spec:
      containers:
        - name: jenkins
          image: orielias/jenkins-image:latest
          ports:
            - containerPort: 8080
          volumeMounts:
            - mountPath: "/var/jenkins_home"
              name: jenkins-home
      volumes:
        - name: jenkins-home
          persistentVolumeClaim:
            claimName: local-pvc

```

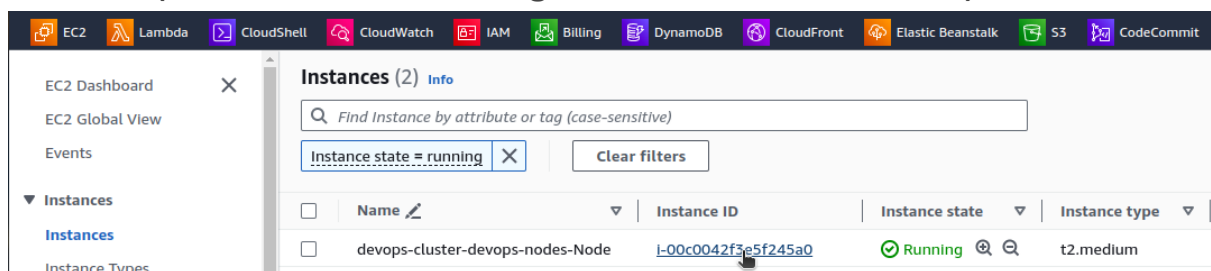
## Apply the Deployment

```
kubectl apply -f jenkins-deployment.yaml
```

the output should be : “deployment.apps/jenkins created”

## Allow Traffic on Port 30000

Navigate to: EC2 -> Instances -> Click on Instance ID (worker node) -> Security -> Security groups -> Inbound rules -> -> Edit inbound rules -> Add rule (custom TCP, Port Range - 30000, CIDR - 0.0.0.0/0) -> Save rules



Details | **Security** | Networking | Storage | Status checks | Monitoring | Tags

▼ Security details

IAM Role  
eksctl-devops-cluster-nodegroup-de-NodeInstanceRole-S0BypvDrCb19

Owner ID  
771593319939

Launch time  
Mon Oct 23 2023 23:02:07 GMT+0300 (Israel Daylight Tin)

Security groups  
sg-00f855da82b0c94dd (eks-cluster-sg-devops-cluster-1624202922)

▼ Inbound rules

Name	Security group rule ID	Port range	Protocol	Source	Security groups
-	sgr-057903ab0e51783a6	All	All	sg-00f855da82b0c94dd	eks-cluster-sg-devops-cluster-1624202922
-	sgr-0b688dc3ba5158d9e	All	All	sg-0b14edcaa5ed6f8b6	eks-cluster-sg-devops-cluster-1624202922

Details | **Inbound rules** | Outbound rules | Tags

Inbound rules (3)

Filter security group rules

<input type="checkbox"/>	Name	Security group rule ID	IP version	Type	Protocol	Port range	Source	Description
<input type="checkbox"/>	-	sgr-057903ab0e51783a6	-	All traffic	All	All	sg-00f855da82b0c94dd	-
<input type="checkbox"/>	-	sgr-0b688dc3ba5158d9e	-	All traffic	All	All	sg-0b14edcaa5ed6f8b6	Allow unman...

Edit inbound rules

Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source	Description - optional	Actions
sgr-057903ab0e51783a6	All traffic	All	All	Custom		Delete
sgr-0b688dc3ba5158d9e	All traffic	All	All	Custom	Allow unmanaged nodes to communicate with control plane (all ports)	Delete
sgr-0b1b166fd3abf4ead	Custom TCP	TCP	30000	Custom	Accessing Jenkins	Delete

Add rule

Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Preview changes Save rules

## Access the Jenkins Website (nodeIP:NodePort)

to find the nodeIP use : `kubectl get nodes -o wide`

copy the EXTERNAL-IP

```
ubuntu@k8s-master:~/ServiceFiles$ kubectl get nodes -o wide
NAME                                STATUS    ROLES    AGE   VERSION   INTERNAL-IP   EXTERNAL-IP
ip-192-168-1-33.ec2.internal        Ready    <none>   16h   v1.27.5-eks-43840fb   192.168.1.33   174.129.2.204
```

now reach the jenkins website through : EXTERNAL-IP:30000

## Retrieve Jenkins Unlock Key

```
kubectl exec -it $(kubectl get pod -n devops -l app=jenkins -o  
jsonpath="{.items[0].metadata.name}") -n devops -- cat  
/var/jenkins_home/secrets/initialAdminPassword
```

copy the output and press 'continue'

Getting Started

## Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

```
/var/jenkins_home/secrets/initialAdminPassword
```

Please copy the password from either location and paste it below.

Administrator password

Continue

## Install plugins

you can always install additional plugins later.

Getting Started

# Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

### Install suggested plugins

Install plugins the Jenkins community finds most useful.

### Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.414.3

# Getting Started

✓ Folders	✓ OWASP Markup Formatter	✓ Build Timeout	✓ Credentials Binding	** Ionicons API
✓ Timestamper	⚙ Workspace Cleanup	⚙ Ant	⚙ Gradle	<b>Folders</b>
⚙ Pipeline	⚙ GitHub Branch Source	⚙ Pipeline: GitHub Groovy Libraries	⚙ Pipeline: Stage View	<b>OWASP Markup Formatter</b>
⚙ Git	⚙ SSH Build Agents	⚙ Matrix Authorization Strategy	⚙ PAM Authentication	** Structs
⚙ LDAP	⚙ Email Extension	⚙ Mailer		** bouncycastle API
				** Instance Identity
				** JavaBeans Activation Framework (JAF) API
				** JavaMail API
				** Pipeline: Step API
				** Token Macro
				<b>Build Timeout</b>
				** Credentials
				** Plain Credentials
				** Trilead API
				** SSH Credentials
				<b>Credentials Binding</b>
				** SCM API
				** Pipeline: API
				** commons-lang3 v3.x Jenkins API
				<b>Timestamper</b>
				** Caffeine API
				** Script Security
				** JAXB
				** SnakeYAML API
				** Jackson 2 API
				** commons-text API
				** Pipeline: Supporting APIs
				** Plugin Utilities API
				** Font Awesome API
				** Bootstrap 5 API
				** JQuery3 API
				** ECharts API
				** - required dependency

Jenkins 2.414.3

## Create Admin user

fill the fields :

## Create First Admin User

Username

Password

Confirm password

Full name

E-mail address



enter the public ip here `http://"public-ip":30000/`

Getting Started

# Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.303.1

Not now

Save and Finish

Getting Started

# Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

Jenkins 2.303.1

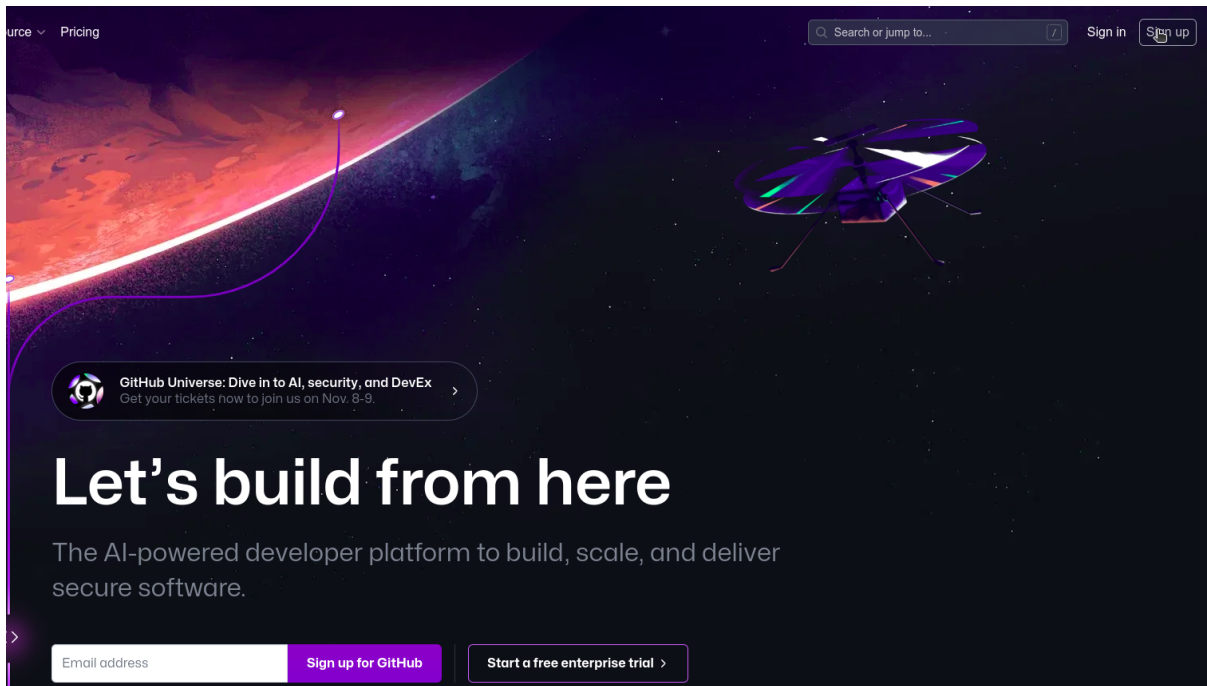
That's it! Jenkins is now ready to use

## **Configure jenkins to work with GitHub :**

### **Create GitHub Account**

go to : <https://github.com>

Click on “Sign Up”



**Fill the Fields:**

Welcome to GitHub!  
Let's begin the adventure

Enter your email\*

✓ orisami233@gmail.com

Create a password\*

✓ .....

Enter a username\*

✓ orioElias

Would you like to receive product updates and  
announcements via email?

Type "y" for yes or "n" for no

✓ n

Verify your account



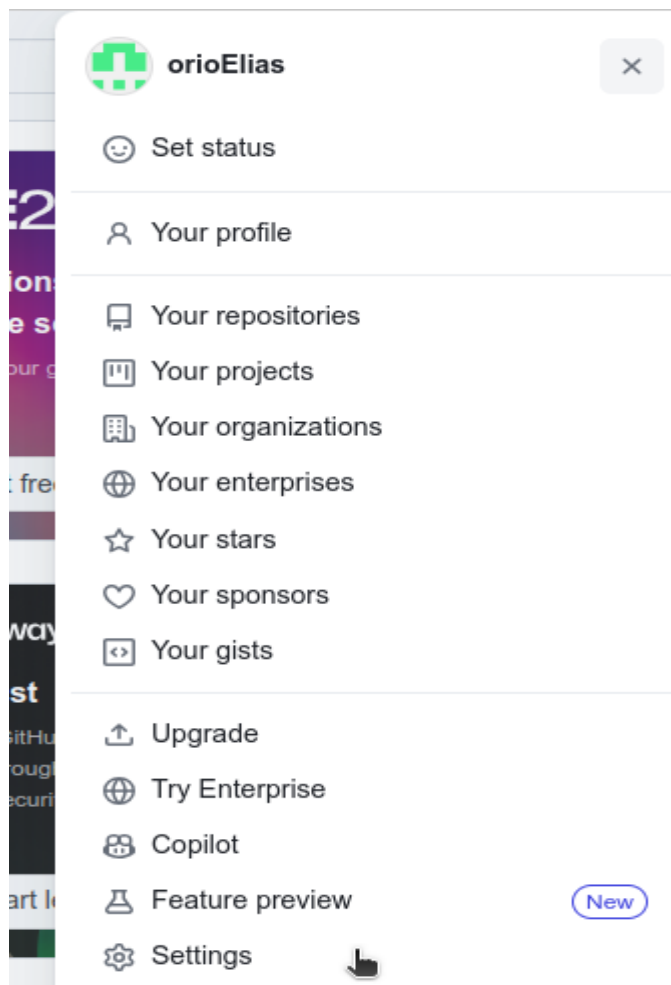
Create account

Copy The Password that sent to your email

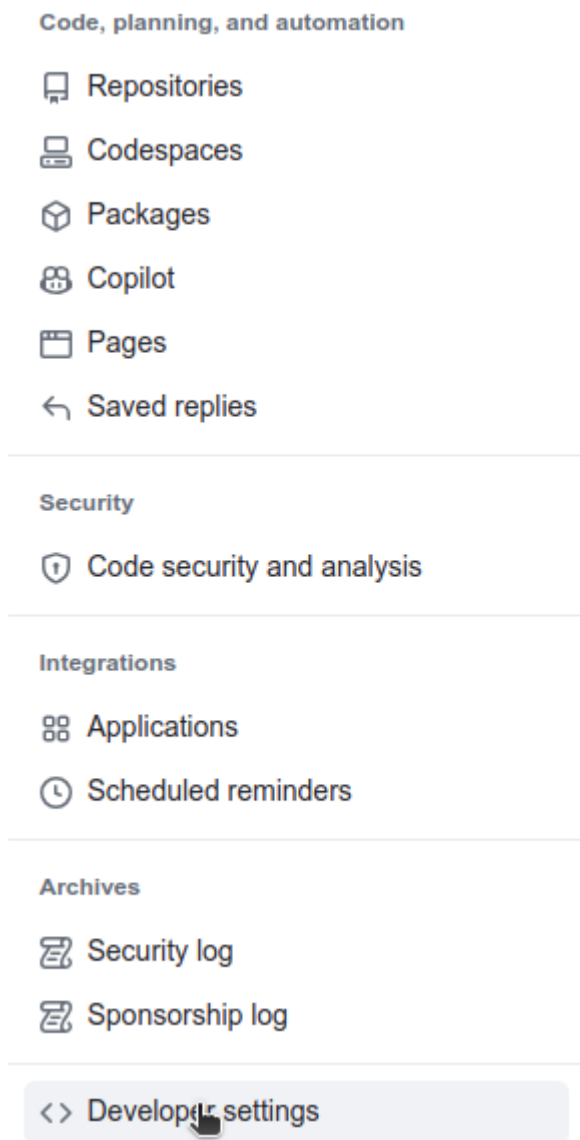


## Create Personal Access Token (PAT) on GitHub:

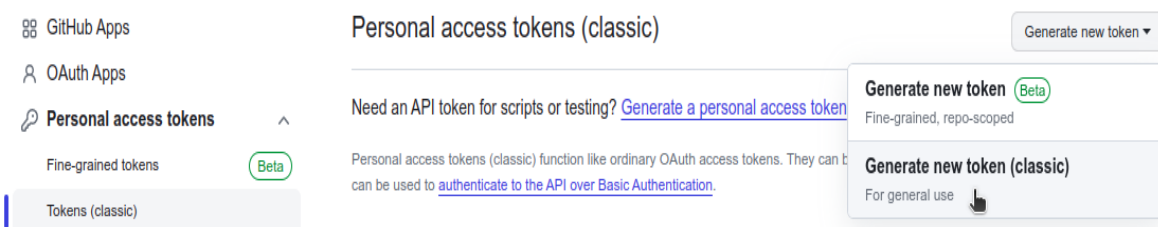
Click on profile picture at top-right and choose "Settings".



Scroll Down And choose "Developer Settings"



Click on "Personal access tokens" from the left sidebar, then click on "Generate new token".



Name the token and give it the necessary scopes (permissions)

## New personal access token (classic)

Personal access tokens (classic) function like ordinary OAuth access tokens. They can be used instead of a password for Git over HTTPS, or can be used to [authenticate to the API over Basic Authentication](#).

### Note

Jenkins Token

What's this token for?

### Expiration \*

90 days

The token will expire on Mon, Jan 22 2024

### Select scopes

Scopes define the access for personal tokens. [Read more about OAuth scopes](#).

<input checked="" type="checkbox"/> <b>repo</b>	Full control of private repositories
<input checked="" type="checkbox"/> repo:status	Access commit status
<input checked="" type="checkbox"/> repo_deployment	Access deployment status
<input checked="" type="checkbox"/> public_repo	Access public repositories
<input checked="" type="checkbox"/> repo:invite	Access repository invitations
<input checked="" type="checkbox"/> security_events	Read and write security events
<input type="checkbox"/> <b>workflow</b>	Update GitHub Action workflows
<input type="checkbox"/> <b>write:packages</b>	Upload packages to GitHub Package Registry
<input type="checkbox"/> read:packages	Download packages from GitHub Package Registry
<input type="checkbox"/> <b>delete:packages</b>	Delete packages from GitHub Package Registry
<input type="checkbox"/> <b>admin:org</b>	Full control of orgs and teams, read and write org projects
<input type="checkbox"/> write:org	Read and write org and team membership, read and write org projects
<input type="checkbox"/> read:org	Read org and team membership, read org projects
<input type="checkbox"/> manage_runners:org	Manage org runners and runner groups
<input type="checkbox"/> <b>admin:public_key</b>	Full control of user public keys
<input type="checkbox"/> write:public_key	Write user public keys
<input type="checkbox"/> read:public_key	Read user public keys
<input checked="" type="checkbox"/> <b>admin:repo_hook</b>	Full control of repository hooks
<input checked="" type="checkbox"/> write:repo_hook	Write repository hooks
<input checked="" type="checkbox"/> read:repo_hook	Read repository hooks
<input checked="" type="checkbox"/> <b>admin:org_hook</b>	Full control of organization hooks

Click "Generate token" at the bottom

Generate token

Cancel

Copy the generated token somewhere safe; you won't be able to see it again.  
(its cut for safety reasons)

Personal access tokens (classic) Generate new token Revoke all

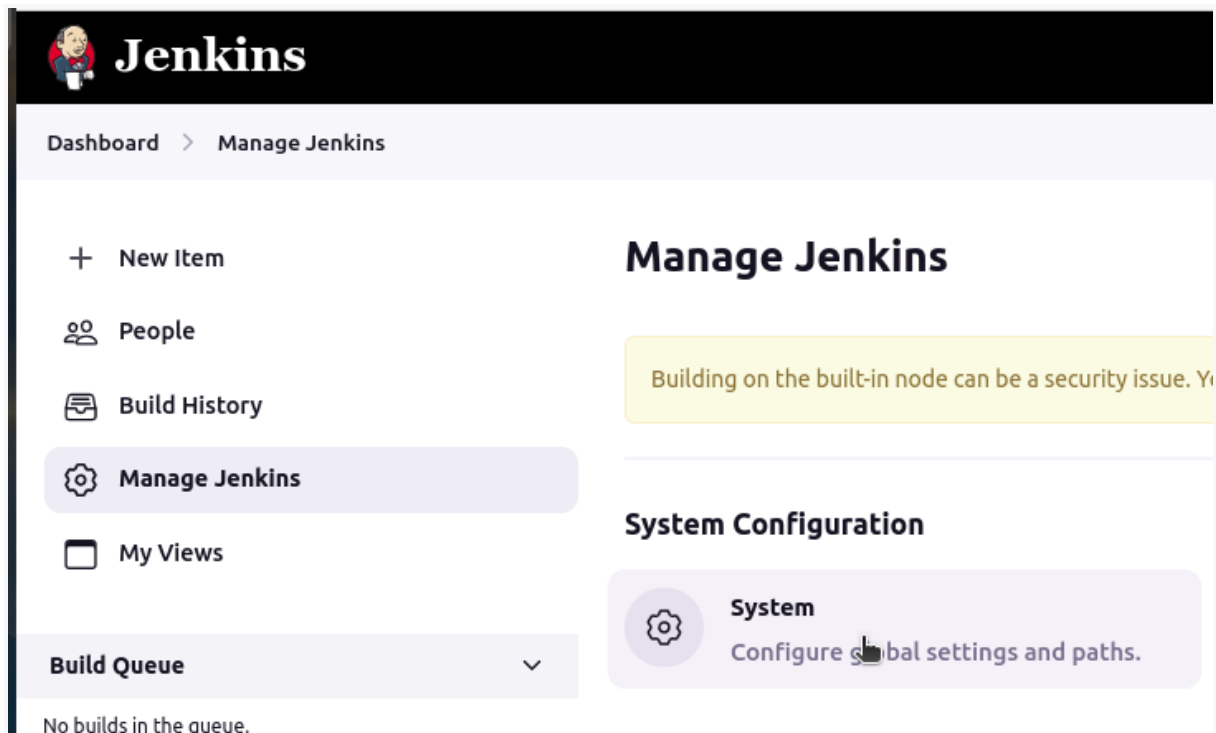
Tokens you have generated that can be used to access the [GitHub API](#).

Make sure to copy your personal access token now. You won't be able to see it again!

`ghs_16P5HsTFer3edV3d1aG1hu31me5eF45C7eD...`

## Add GitHub Configuration In Jenkins :

Open Jenkins in your browser, then navigate to "Manage Jenkins" -> "System."



Scroll down until you find the GitHub section

Click on "Add GitHub Server" and fill in the details.

- Name: Can be any name to identify this GitHub configuration.
- API URL: Usually this is `https://api.github.com` for GitHub.com

The screenshot shows the 'Add GitHub Server' form in Jenkins. The form is titled 'GitHub Server' and has a red 'X' icon in the top right corner. It contains the following fields:

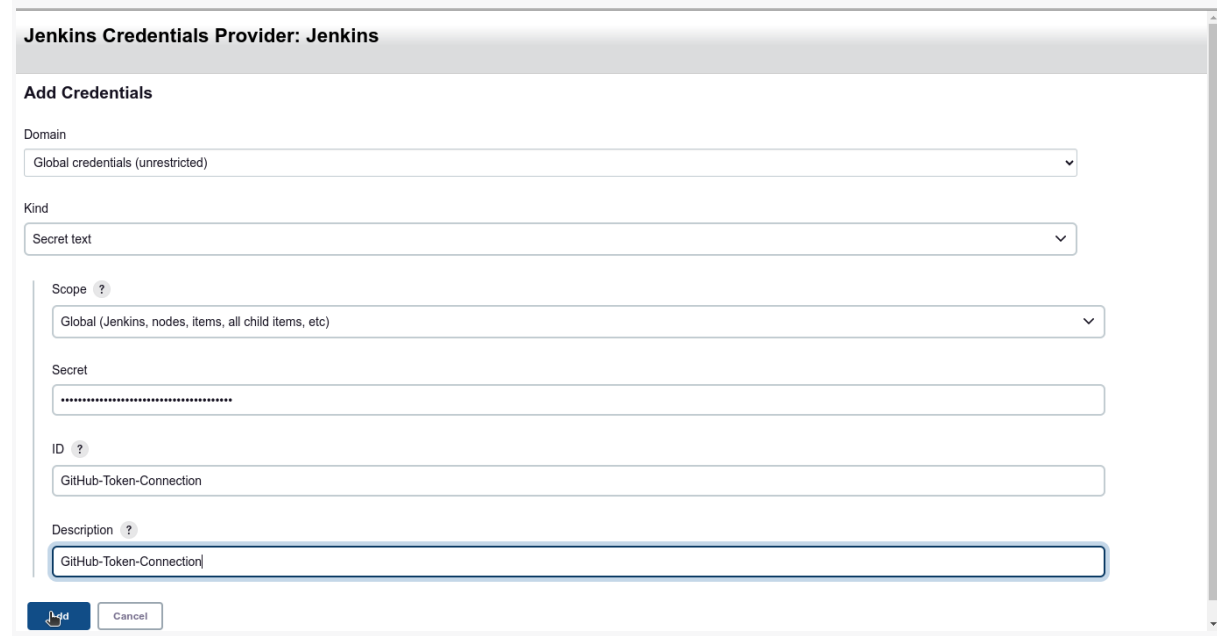
- Name:** A text input field with the value 'GitHub Connection'.
- API URL:** A text input field with the value 'https://api.github.com'.
- Credentials:** A dropdown menu with the value '- none -'.

Below the form is a 'Test connection' button. At the bottom of the form, there is a checkbox labeled 'Manage hooks' which is currently unchecked. A 'Jenkins Credentials Provider' tooltip is visible over the 'Credentials' dropdown.



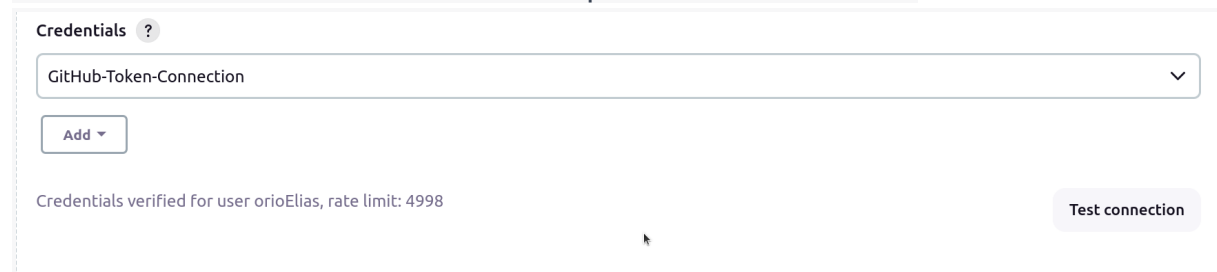
For "Credentials," click the "Add" button next to the dropdown.

- Kind: Choose "Secret text."
- Secret: Paste your GitHub Personal Access Token here.
- ID: Can be any name to identify this secret.
- Description: Optional, but helps you remember what this secret is for.



The screenshot shows the 'Jenkins Credentials Provider: Jenkins' interface. At the top, there's a header 'Jenkins Credentials Provider: Jenkins'. Below it, the section 'Add Credentials' is active. The form contains several fields: 'Domain' with a dropdown menu showing 'Global credentials (unrestricted)'; 'Kind' with a dropdown menu showing 'Secret text'; 'Scope' with a dropdown menu showing 'Global (Jenkins, nodes, items, all child items, etc)'; 'Secret' with a text input field containing a masked string of asterisks; 'ID' with a text input field containing 'GitHub-Token-Connection'; and 'Description' with a text input field containing 'GitHub-Token-Connection'. At the bottom left, there are two buttons: 'Add' (highlighted in blue) and 'Cancel'.

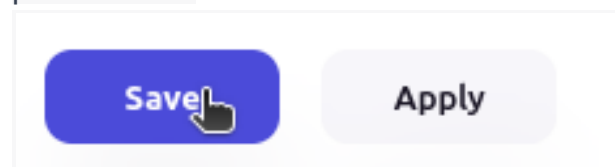
choose the new created credentials and press "Test connection"



The screenshot shows a dropdown menu for 'Credentials' with a question mark icon. The dropdown is open, showing a single item 'GitHub-Token-Connection' with a downward arrow. Below the dropdown, there is an 'Add' button with a downward arrow. At the bottom of the panel, there is a status message: 'Credentials verified for user orioElias, rate limit: 4998'. To the right of this message is a button labeled 'Test connection'.

the output should be something like "Credentials verified for user orioElias, rate limit:4999"

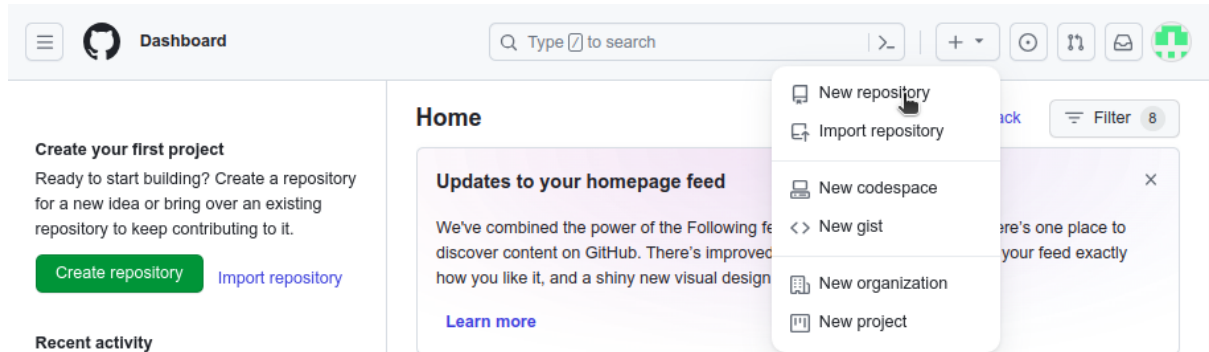
press "Save"



The screenshot shows two buttons side-by-side. The first button is blue with the text 'Save' and a hand cursor icon pointing at it. The second button is light gray with the text 'Apply'.

## Create a GitHub Repository

log in GitHub, you'll see a "+" icon in the upper-right corner next to your profile picture. Click on it and select "New repository."




## Fill your repository details, then Click "Create repository"

### Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

*Required fields are marked with an asterisk (\*).*


Owner \*

 orioElias

/

Repository name \*


K8S-Project

 K8S-Project is available.


Great repository names are short and memorable. Need inspiration? How about [effective-succotash](#) ?

Description (optional)

K8S Project Integrated With Jenkins

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more about READMEs.](#)

Add .gitignore


.gitignore template: None


Choose which files not to track from a list of templates. [Learn more about ignoring files.](#)

Choose a license

License: None

A license tells others what they can and can't do with your code. [Learn more about licenses.](#)

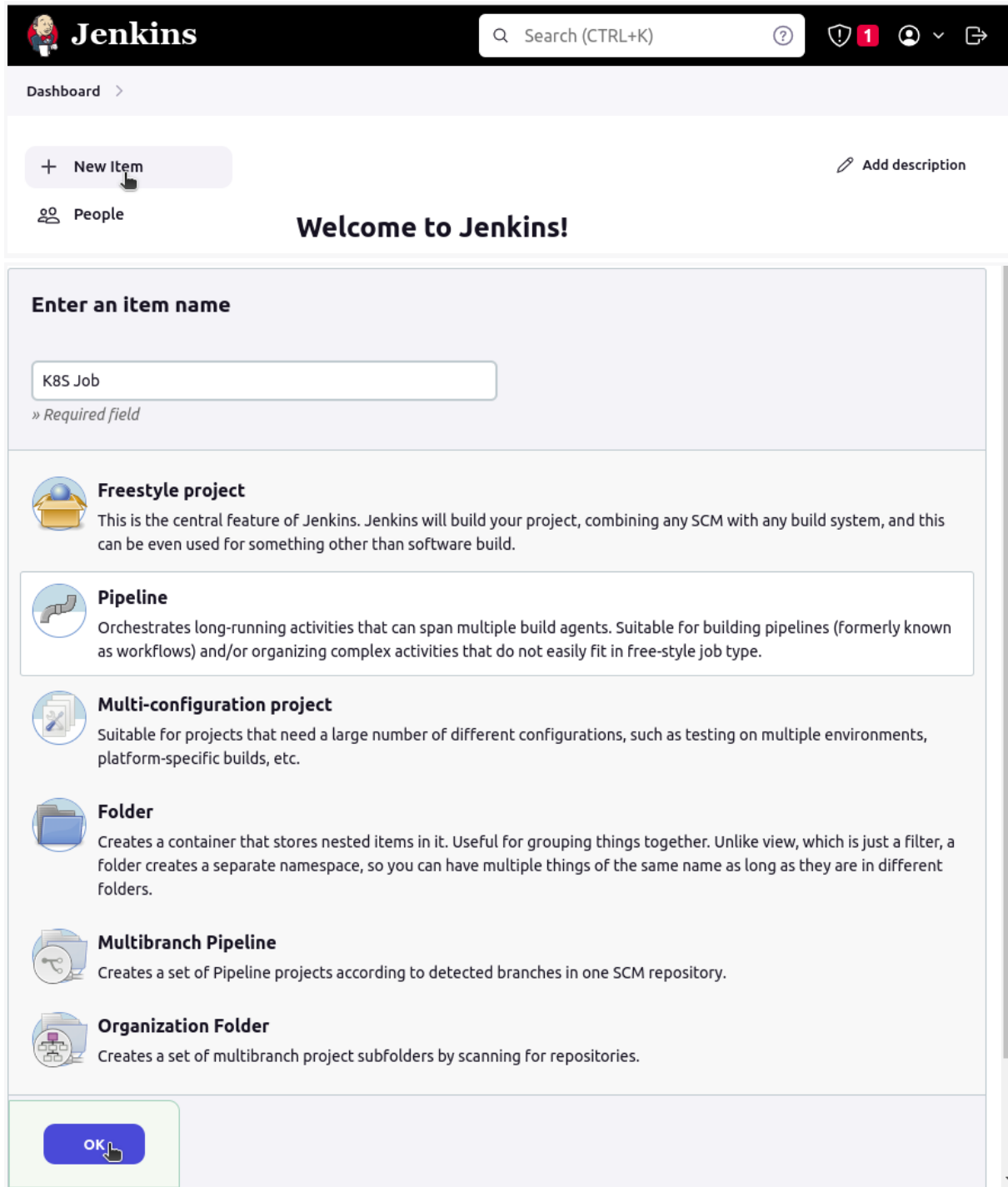
This will set  main as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository

## Connect the New GitHub Repository To Jenkins:

Open your Jenkins interface, Click on "New Item" on the Jenkins dashboard, select "Pipeline," and then proceed.



**Jenkins** Search (CTRL+K) ? 1

Dashboard >

+ New Item Add description

People

### Welcome to Jenkins!

**Enter an item name**

K8S Job

» Required field

- Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
- Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
- Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
- Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.
- Organization Folder**  
Creates a set of multibranch project subfolders by scanning for repositories.

OK

In the pipeline configuration, Check "GitHub hook trigger for GITScm polling". Additionally you'll need to specify the source as your GitHub repository.

- **Source:** Git
- **Repository URL:** The URL of the GitHub repository you just created
- **Credentials:** Create a new set of credentials in Jenkins using the same GitHub Personal Access Token (PAT) you generated earlier.

**Configure**

- General
- Advanced Project Options**
- Pipeline

**Pipeline**

Definition

Pipeline script from SCM

SCM ?

Git

Repositories ?

Repository URL ?

https://github.com/orioElias/K8S-Project

Credentials ?

- none -

Jenkins Credentials Provider

Jenkins

Advanced

Add Repository

Branches to build ?

In the "Credentials" dropdown, click on the "Add" button and Jenkins.

- For the "Kind" field, choose "Username with password."  
Enter your GitHub username in the "Username" field.
- Paste your GitHub Personal Access Token in the "Password" field.
- Add an ID and Description to identify these credentials.

Global credentials (unrestricted)

Kind  
Username with password

Scope ?  
Global (Jenkins, nodes, items, all child items, etc)



Username ?  
orioElias

☐ Treat username as secret ?

Password ?  
\*\*\*\*\*

ID ?  
GitHub-Username-Password-Credentials

Description ?  
GitHub Username Password Credentials



Now, choose the New Credentials, Change “Branch Specifier (blank for 'any')”  
From: “\*/master” To “\*/main” And Click “SAVE”


SCM ?  
Git

Repositories ?

Repository URL ?  
https://github.com/orioElias/K8S-Project


Credentials ?  
orioElias/\*\*\*\*\* (GitHub Username Password Credentials)





Branches to build ?

Branch Specifier (blank for 'any') ?  
\*/main

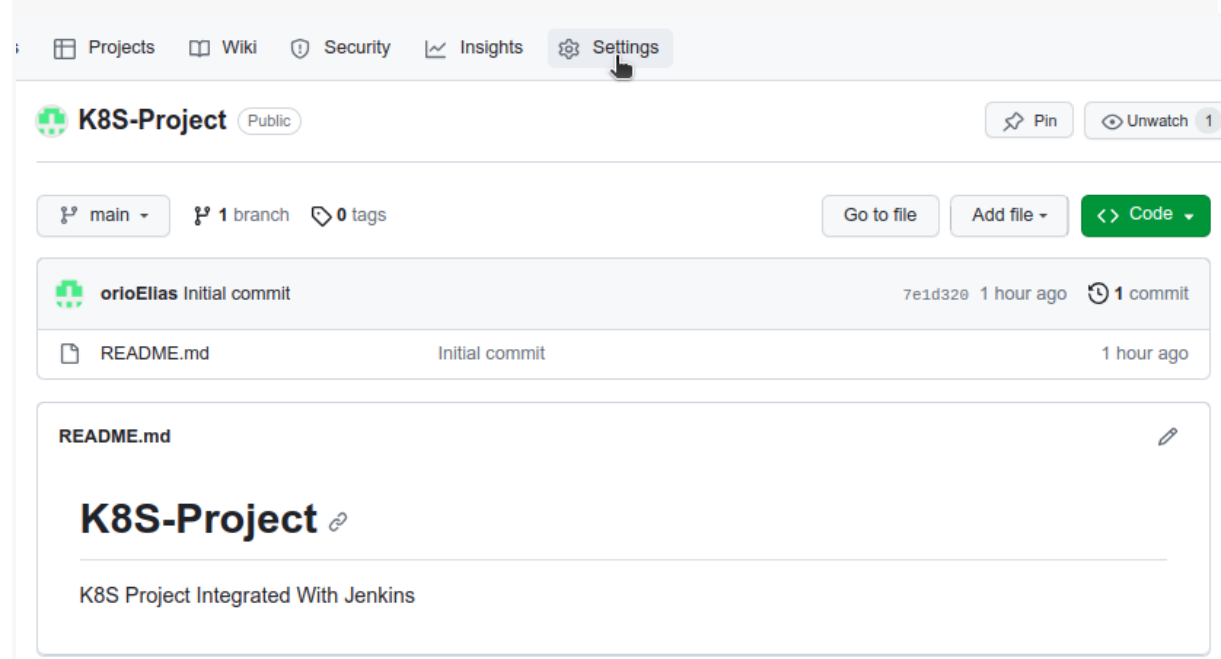


Repository browser ?

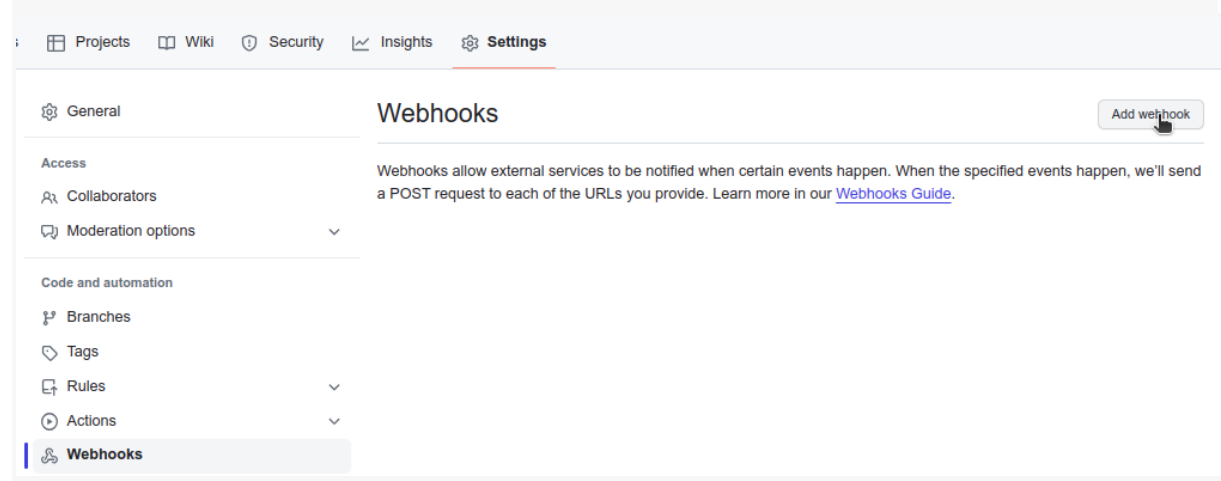
 

## Add GitHub Webhook to Trigger Jenkins Pipeline

GitHub Repository Settings: Navigate to "Settings" on your GitHub repository.



Webhooks: Go to the "Webhooks" tab and click on "Add webhook."



Use your Jenkins URL followed by `/github-webhook/`. For example, `http://your-public-ip:30000/github-webhook/`

Content type: Choose `application/json`

Secret: Leave this blank if you haven't set up a secret in Jenkins for webhook validation.

Choose "Just the push event" if you want the pipeline to run only when new commits are pushed.

Active: Make sure this checkbox is checked.

Click: "Add webhook"

**Webhooks / Add webhook**

We'll send a `POST` request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, `x-www-form-urlencoded`, *etc*). More information can be found in [our developer documentation](#).

---

**Payload URL \***

**Content type**

**Secret**

**Which events would you like to trigger this webhook?**

☒ Just the push event.

☐ Send me **everything**.


☐ Let me select individual events.

---

☒ **Active**

We will deliver event details when this hook is triggered.

**Add webhook**



## Allow Traffic on Port 80 (For the Webhook, Same as we did Before)

Here's how to add a rule for port 80 in your AWS Security Group:


Navigate to: EC2 -> Instances -> Click on Instance ID (worker node) -> Security -> Security groups -> Inbound rules -> -> Edit inbound rules -> Add rule (HTTP, CIDR - 0.0.0.0/0) -> Save rules

### Edit inbound rules Info

Inbound rules control the incoming traffic that's allowed to reach the instance.

#### Inbound rules Info

Security group rule ID	Type <small>Info</small>	Protocol <small>Info</small>	Port range <small>Info</small>	Source <small>Info</small>	Description - optional <small>Info</small>		
sg-057903ab0e51783a6	All traffic	All	All	Custom	<input type="text" value="Q"/>	<input type="text" value=""/>	<input type="button" value="Delete"/>
				<input type="text" value="sg-00f855da82b0c94dd"/>			
sg-0b688dc3ba5158d9e	All traffic	All	All	Custom	<input type="text" value="Q"/>	Allow unmanaged nodes to communic...	<input type="button" value="Delete"/>
				<input type="text" value="sg-0b14edcaa5ed6f8b6"/>	Allow unmanaged nodes to communicate with control plane (all ports)		
sg-0b1b166fd3abf4ead	Custom TCP	TCP	30000	Custom	<input type="text" value="Q"/>	Accessing Jenkins	<input type="button" value="Delete"/>
				<input type="text" value="0.0.0.0/0"/>			
-	HTTP	TCP	80	Anywhere...	<input type="text" value="Q"/>	GitHub Webhook	<input type="button" value="Delete"/>
				<input type="text" value="0.0.0.0/0"/>			

 Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.



## Clone the repository to your local machine && Checking the Webhook

run in the terminal (in the directory that you want to clone your repo) :  
git clone <https://github.com/><your-username>/<directory-name>.git

this will create a new directory that contains your GitHub repository.

```
ori@OriElias:~/Desktop/k8s project$ git clone https://github.com/orioElias/K8S-Project.git
Cloning into 'K8S-Project'...
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
```

cd <directory-name>

Make some changes to the existing files or add new files to the directory.

git add <filename>

git commit -m "Your commit message here"

git config --global credential.helper 'cache --timeout=3600' (optional)

git push origin main

username: <your-username>

password: <Personal Access Token>

After you push the commit, your GitHub webhook should trigger your Jenkins pipeline if it's configured correctly.

## Download an ASP.NET Core web application.

**Download from Official Samples:** Microsoft offers various sample projects that you can download directly. You can find these on the official [ASP.NET Core samples page](#).

## Install .NET SDK plugin

**Plugins**

Search: SDK

Name ↓	Enabled
<b>.NET SDK Support</b> 1.4.0 This plugin provides handling for <a href="#">.NET SDKs</a> (Core and 5.0+) as global tools in Jenkins. Includes a build wrapper and several convenience builders. <a href="#">Report an issue with this plugin</a>	<input checked="" type="checkbox"/>

- Click on "Manage Jenkins" on the left sidebar.
- Then click on "Tools"

## Manage Jenkins

Building on the built-in node can be a security issue. You should set up distributed builds. See [the documentation](#).

### System Configuration



#### System

Configure global settings and paths.



#### Tools

Configure tools, their locations and automatic installers.

## Configure as your needed

Name ?

MyDotNetSDK

☐ Telemetry Opt-Out ?

☒ Install automatically ?

Install from microsoft.com ?

Label ?

.NET Version ?

.NET 6.0 - Status Unknown (end of support: 2024-11-12)

☒ Include Previews ?

Release ?

6.0.24, released 2023-10-24 (includes security fixes)

Release Notes

SDK ?

6.0.416

C# 12.0, F# 6.0, VB 16.9

Platform ?

linux-x64 (Linux - x64)

Direct Download

Save

Apply

## Create Role Files

### deployment-manager-role.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: deploy
  name: deployment-manager
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "create", "update", "delete"]
```

## deployment-manager-rolebinding.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: deployment-manager-binding
  namespace: deploy
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: deployment-manager
subjects:
- kind: ServiceAccount
  name: default
  namespace: devops
```

## Create a Jenkinsfile


```
1 pipeline {
2   agent any
3
4   tools {
5     // Define the .NET SDK tool
6     dotnetsdk 'MyDotNetSDK'
7   }
8
9   environment {
10    // Add this line to disable globalization
11    SYSTEM_GLOBALIZATION_INVARIANT = 'true'
12  }
13
14  stages {
15    stage('Checkout') {
16      steps {
17        checkout scm
18      }
19    }
20
21    stage('Docker Build') {
22      steps {
23        sh '''
24          docker build -t my-dotnet-app:latest ./MySimpleWebApp
25        '''
26      }
27    }
28
29    stage('Docker Push') {
30      steps {
31        sh '''
32          docker tag my-dotnet-app:latest orielias/my-dotnet-app:latest
```


```

stage('Deploy to Deploy Namespace') {
    steps {
        sh '''
            if ! kubectl get namespaces | grep -q 'deploy'; then
                kubectl create namespace deploy
            fi
            kubectl apply -f deployment-deploy.yaml --namespace=deploy
        '''
    }
}
}

```

## Create Dockerfile

 Dockerfile ×

home > ori > Desktop > k8s project > K8S-Project >  Dockerfile

```

1  # Use the SDK image to build the app
2  FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS build
3  WORKDIR /source
4
5  # Restore and publish the app
6  COPY MySimpleWebApp/MySimpleWebApp.csproj .
7  RUN dotnet restore
8  COPY MySimpleWebApp/. .
9  RUN dotnet publish -c release -o /app
10
11 # Use the runtime image to run the app
12 FROM mcr.microsoft.com/dotnet/aspnet:6.0 AS runtime
13 WORKDIR /app
14 COPY --from=build /app .
15 ENTRYPOINT ["dotnet", "MySimpleWebApp.dll"]
16

```

## Create deployment-deploy.yaml file

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: dotnetcore-deploy
5    namespace: deploy
6  spec:
7    replicas: 2
8    selector:
9      matchLabels:
10       app: dotnetcore-deploy
11    template:
12      metadata:
13        labels:
14          app: dotnetcore-deploy
15      spec:
16        containers:
17          - name: dotnetcore-deploy
18            image: orielias/my-dotnet-app:latest
19            ports:
20              - containerPort: 80
21
```

## Create a service To Access this WEB APP

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: dotnetcore-service
5    namespace: deploy
6  spec:
7    type: NodePort
8    selector:
9      app: dotnetcore-deploy
10   ports:
11     - protocol: TCP
12       port: 80
13       targetPort: 80
14       nodePort: 30080
15
```

Your file structure suppose to be similar to this :

```
|— deployment-deploy.yaml
|— Dockerfile
|— dotnetcore-deploy.yaml
|— Jenkinsfile
|— MySimpleWebApp
|   |— mySimpleWebApp.csproj
|   |— program.cs
|   |— startup.cs
```

Now you can push the new Files to the repository and See the pipeline runs :

Declarative: Checkout SCM	Declarative: Tool Install	Checkout	Docker Build	Docker Push	Deploy to Devops Namespace	Deploy to Deploy Namespace
637ms	220ms	636ms	742ms	127ms	83ms	77ms
