

[Open in app](#)[Sign up](#)[Sign In](#)Gauri Shirkande · [Follow](#)

4 min read · Mar 6

[Listen](#)[Share](#)

HashiCorp

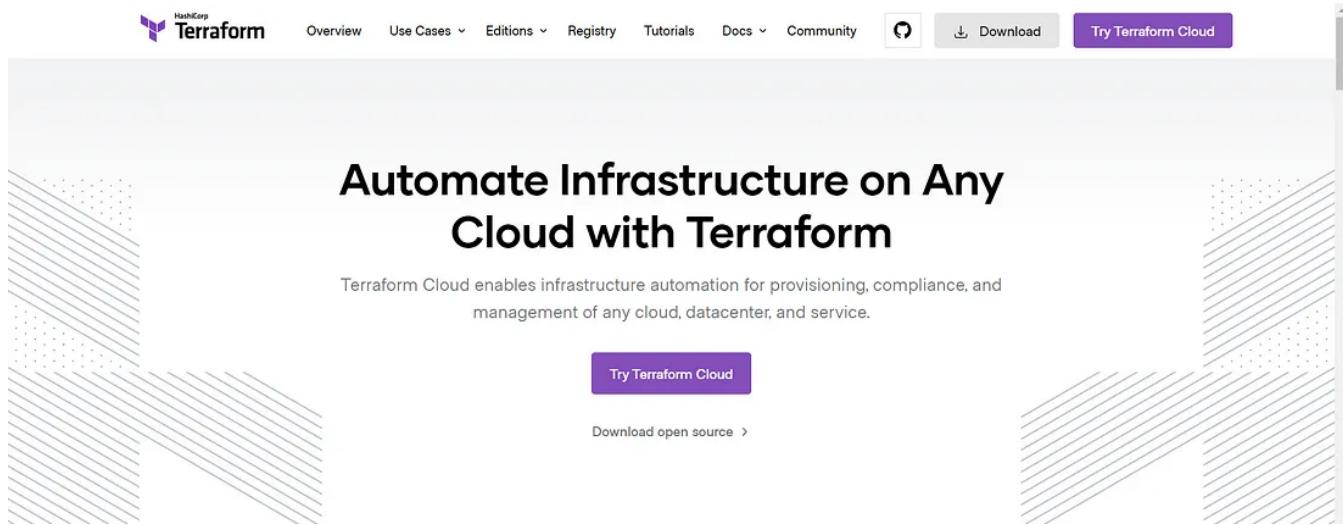
**Terraform**

Terraform is an open-source infrastructure as a code tool that lets you build, change, and version cloud and on-prem resources safely and efficiently. It helps us manage and provision the infrastructure using a simple language which is Hashicorp Configuration Language. Terraform being cloud-agnostic can be used to automate the provisioning and managing of resources.

In this blog, we'll see how to create an AWS EC2 Instance using Terraform. So let's begin.

## 1. Download Terraform on a Windows machine

Go to the official website of terraform: <https://www.terraform.io/>.

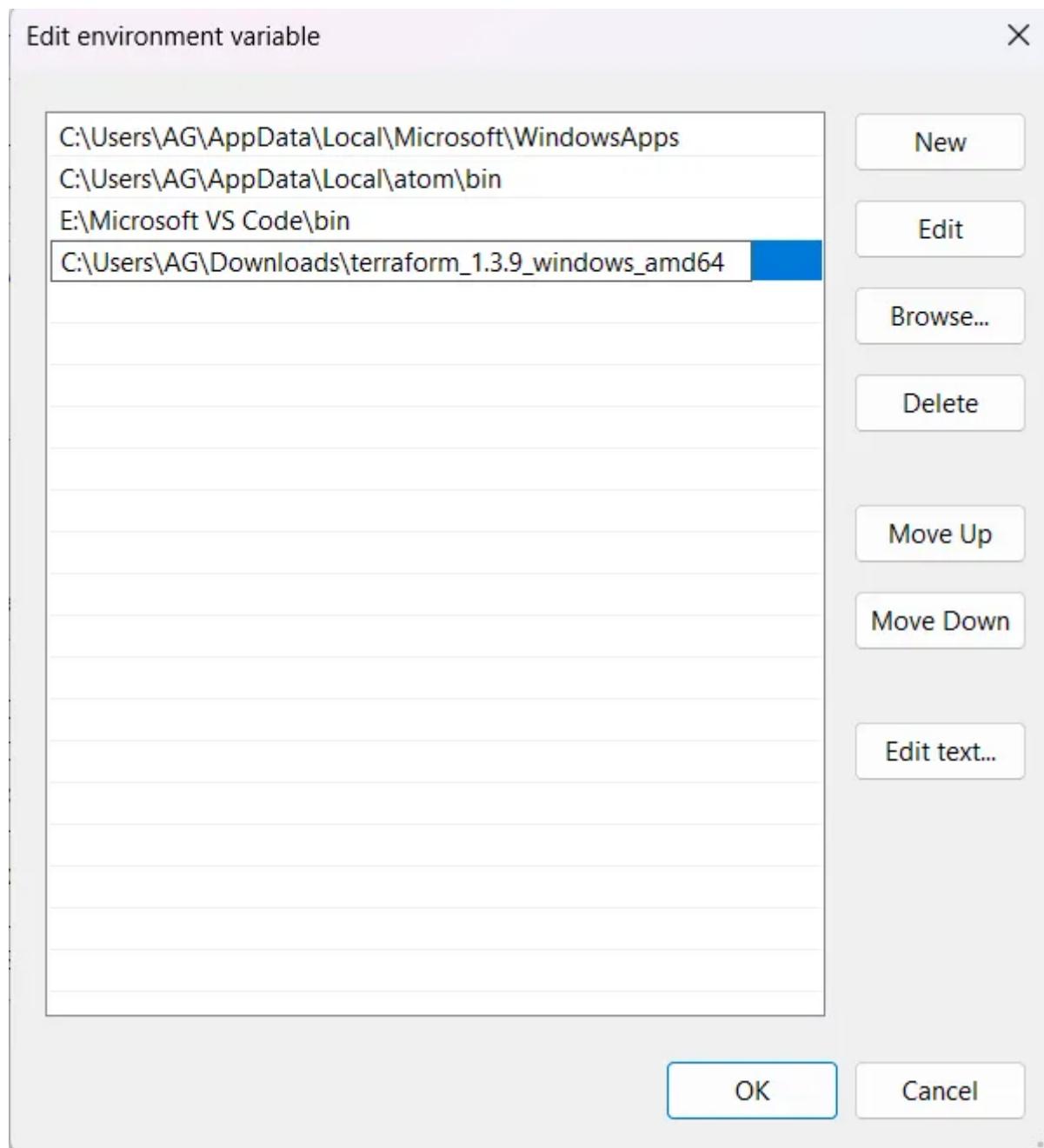


Click on **Download** and then select the **Windows** tab since we are downloading for a windows machine.

And then select the AMD64 binary for downloading terraform.

Once you click on Download, a zip folder for terraform gets downloaded on the local windows machine.

Unzip the folder and then add the path of your binary to the Environment Variables.



Once done, open the command prompt and check the version of installed terraform by using the command below:

```
terraform version
```

## 2. Download and Install AWS-CLI on windows

Download and run the AWS CLI MSI installer for Windows:

<https://awscli.amazonaws.com/AWSCLIV2.msi>

Once done, open the command prompt and check the version of the installed AWS CLI by using the command below:

```
aws --version
```

### 3. Create an AWS User with the required permission

Go to your AWS Management Console and in services search for IAM. In the left side panel, select **Users**. In the **Users** section, click on **Add Users**.

The screenshot shows the AWS IAM service in the AWS Management Console. The left sidebar has 'Identity and Access Management (IAM)' selected. The main area shows a list of users with one entry: 'athu' (User name: athu, Groups: admin and junior-devs, Last activity: 105 days ago, MFA: None, Password age: 105 days ago). There are buttons for 'Delete' and 'Add users'.

Enter the **User name** and click **Next**

The screenshot shows the 'Create user' wizard Step 1: Specify user details. It asks for a 'User name' (gauris) and provides optional console access. A note says you can generate programmatic access later. Buttons for 'Cancel' and 'Next' are at the bottom.

Select **Attach policies directly** option and select the **AdministratorAccess** or **AmazonEC2FullAccess** and click **Next** again.

**Set permissions**

Add user to an existing group or create a new one. Using groups is a best-practice way to manage user's permissions by job functions. [Learn more](#)

**Permissions options**

- Add user to group Add user to an existing group, or create a new group. We recommend using groups to manage user permissions by job function.
- Copy permissions Copy all group memberships, attached managed policies, and inline policies from an existing user.
- Attach policies directly Attach a managed policy directly to a user. As a best practice, we recommend attaching policies to a group instead. Then, add the user to the appropriate group.

**Permissions policies (1/1081)**  
Choose one or more policies to attach to your new user.

| Policy name   | Type                       | Attached entities |
|---|----------------------------|-------------------|
| <input checked="" type="checkbox"/> AccessAnalyzerServiceRolePolicy | AWS managed                | 0                 |
| <input checked="" type="checkbox"/> AdministratorAccess             | AWS managed - job function | 1                 |
| <input type="checkbox"/> AdministratorAccess-Amplify                | AWS managed                | 0                 |

Then Review the User details and click on **Create User**.

After this, you will see the user created.

**gauris**

**Summary**

|  |                            |                             |
|--|----------------------------|-----------------------------|
| ARN<br><a href="#">arn:aws:iam::094298998682:user/gauris</a> | Console access<br>Disabled | Access key 1<br>Not enabled |
| Created<br>March 05, 2023, 20:27 (UTC+05:30)                 | Last console sign-in<br>-  | Access key 2<br>Not enabled |

**Permissions**

**Permissions policies (1)**  
Permissions are defined by policies attached to the user directly or through groups.

| Policy name                                  | Type                       | Attached via |
|--|----------------------------|--------------|
| <input type="checkbox"/> AdministratorAccess | AWS managed - job function | Directly     |

Then within the user select **Security credentials**.

The screenshot shows the AWS IAM 'Users' section for a user named 'gauris'. The 'Security credentials' tab is selected. It lists two access keys: 'Access key 1' and 'Access key 2', both of which are currently disabled. The ARN for the user is listed as arn:aws:iam::094298998682:user/gauris. The user was created on March 05, 2023, at 20:27 (UTC+05:30).

Create an Access Key for the user by selecting the **Create access key**.

The screenshot shows the 'Access keys' section for the 'gauris' user. There are no existing access keys. A button labeled 'Create access key' is visible. A note above the button states: 'No MFA devices. Assign an MFA device to improve the security of your AWS environment'.

Then select the **Command Line Interface (CLI)** and click **Next**.

Step 1  
Access key best practices & alternatives

Step 2 - optional  
Set description tag

Step 3  
Retrieve access keys

**Command Line Interface (CLI)**  
You plan to use this access key to enable the AWS CLI to access your AWS account.

**Local code**  
You plan to use this access key to enable application code in a local development environment to access your AWS account.

**Application running on an AWS compute service**  
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.

**Third-party service**  
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.

**Application running outside AWS**  
You plan to use this access key to enable an application running on an on-premises host, or to use a local AWS client or third-party AWS plugin.

Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

Then Retrieve access keys by downloading the access and secret keys.

Step 1  
Access key best practices & alternatives

Step 2 - optional  
Set description tag

Step 3  
Retrieve access keys

**Access key**  
If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

| Access key           | Secret access key               |
|----------------------|---------------------------------|
| AKIARL5FIQ6NHKIEHBE4 | XXXXXXXXXX <a href="#">Show</a> |

**Access key best practices**

- Never store your access key in plain text, in a code repository, or in code.
- Disable or delete access key when no longer needed.
- Enable least-privilege permissions.
- Rotate access keys regularly.

For more details about managing access keys, see the [Best practices for managing AWS access keys](#).

[Download .csv file](#) [Done](#)

Feedback Language © 2023, Amazon Web Services India Private Limited or its affiliates. Privacy Terms Cookie preferences

## 4. Configure AWS CLI

Open your Command Prompt and run aws configure

```
aws configure
```

It will prompt you to enter the Access and Secret keys along with the region and format.

```
AWS Access Key ID [None]: <Access Key ID>
AWS Secret Access Key [None]: <Secret Key ID>
Default region name [None]: <Region>
Default output format [None]: <format>
```

## 5. Create your terraform configuration file

Create a terraform configuration file with .tf extension in HCL (Hashicorp Configuration Language). For creating an EC2 instance we need to mention the AWS provider in the configuration file. So we first define the provider and then define the resource that is an ec2 instance with required parameters.

```
provider "aws" {
    region = "us-east-1"
}

resource "aws_instance" "myInstance" {
    ami           = "ami-006dcf34c09e50022"
    instance_type = "t2.micro"

    tags = {
        Name = "MyInstance"
    }
}
```

## 6. Initialize the terraform working directory

Before actually creating the infrastructure we first need to initialize the working directory. This can be done using **terraform init** command. On running this command, it downloads all the required plugins necessary for building your infrastructure.

```
terraform init
```

In order to validate the configuration file run the command below:

```
terraform validate
```

## 7. Preview the changes with terraform plan

Once done with the initialization and validation of the configuration file, run the command below to preview the changes before actually implementing them on AWS.

```
terraform plan
```

## 8. Creating the infrastructure using terraform apply

Finally, run the command **terraform apply** to actually deploy the infrastructure on the AWS cloud. This command creates the infrastructure that was previously planned. With this command, **terraform.tfstate** file gets created which keeps the track of resources created using terraform. This file is by default stored locally, but it can also be stored remotely, which works better in a team environment.

```
terraform apply
```

On executing this command it asks for the re-confirmation. Once you enter **yes**, it implements the desired changes. You can see the instance running in your AWS Management Console after the process of creation is completed.

In this way, we can create an EC2 instance using Terraform.

Hope you found this blog useful. Happy Learning !!

[Follow](#)

## Written by Gauri Shirkande

14 Followers

---

### More from Gauri Shirkande



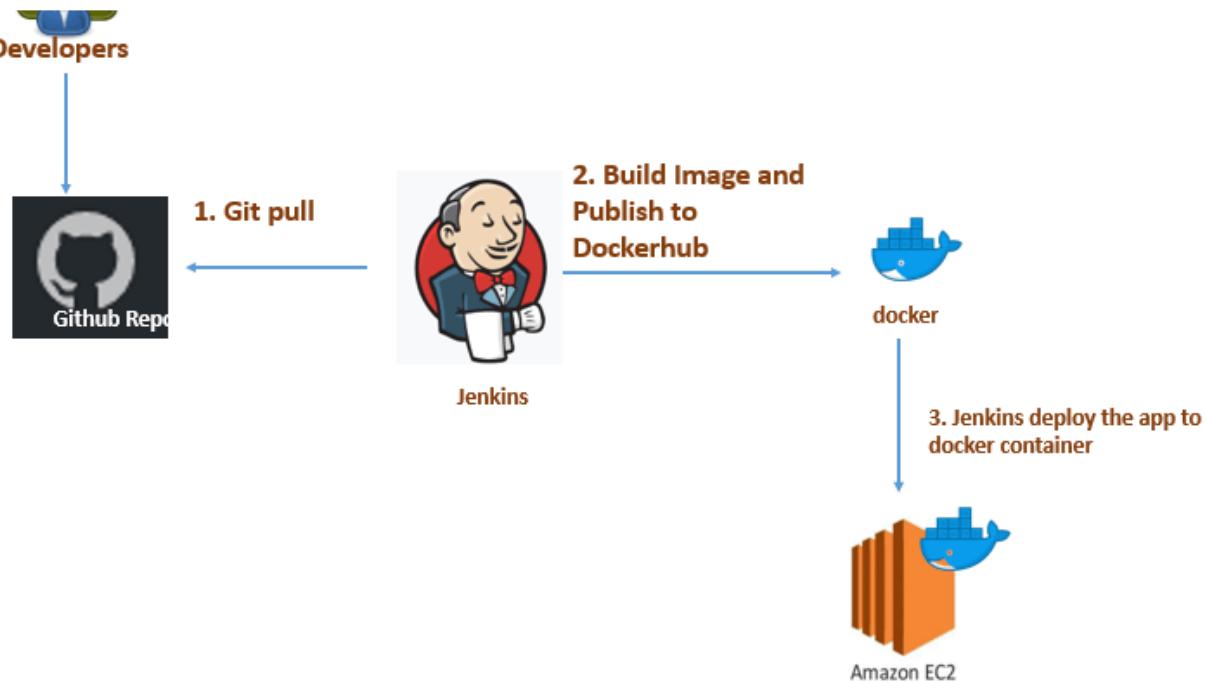
Gauri Shirkande

## Understanding Kubernetes Architecture and Components, Installation and Configuration

What is Kubernetes?

6 min read · Apr 25





 Gauri Shirkande

## Create a pipeline using Jenkins, and GitHub and deploy it to AWS EC2

In this article, we are going to create a CI/CD pipeline using Jenkins for a NodeJS project.

5 min read · Mar 16



 Gauri Shirkande

## Terraform—Best practices

Terraform is one of the most popular IaC (Infrastructure as a Code) tools that automates the creation of infrastructure using code.

4 min read · Mar 25



*Reusable, composable, battle-tested*  
**TERRAFORM  
MODULES**



Gauri Shirkande

## Introduction to Terraform Modules

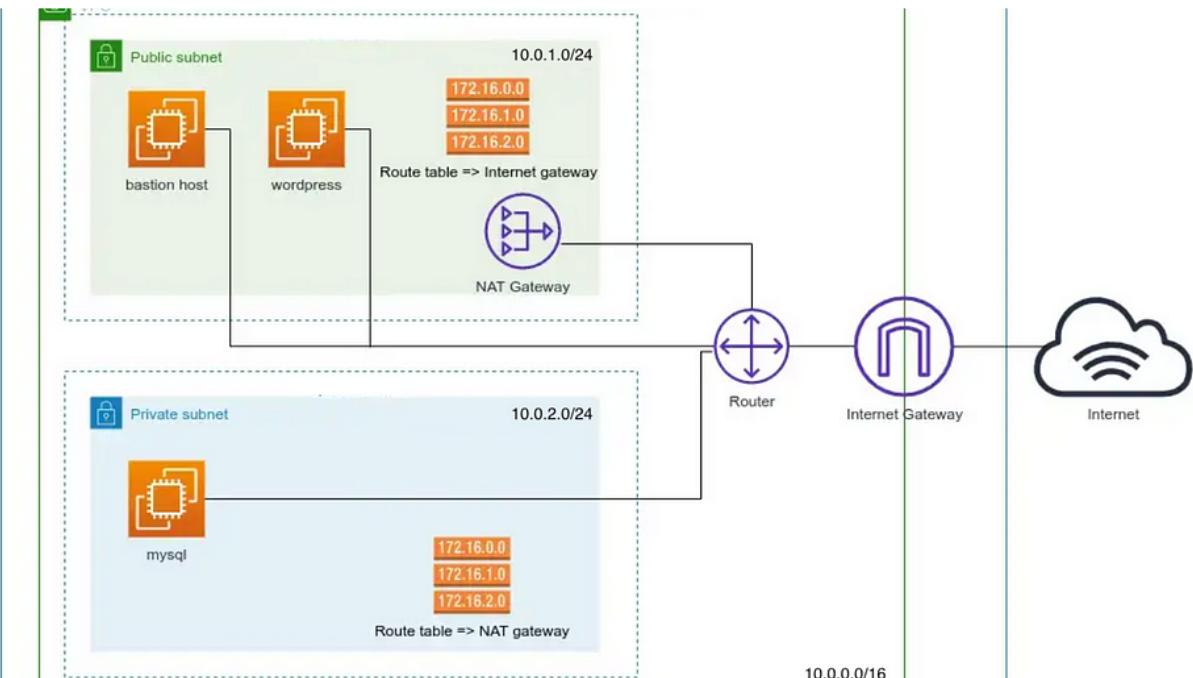
What are Terraform Modules?

3 min read · Mar 14



See all from Gauri Shirkande

## Recommended from Medium



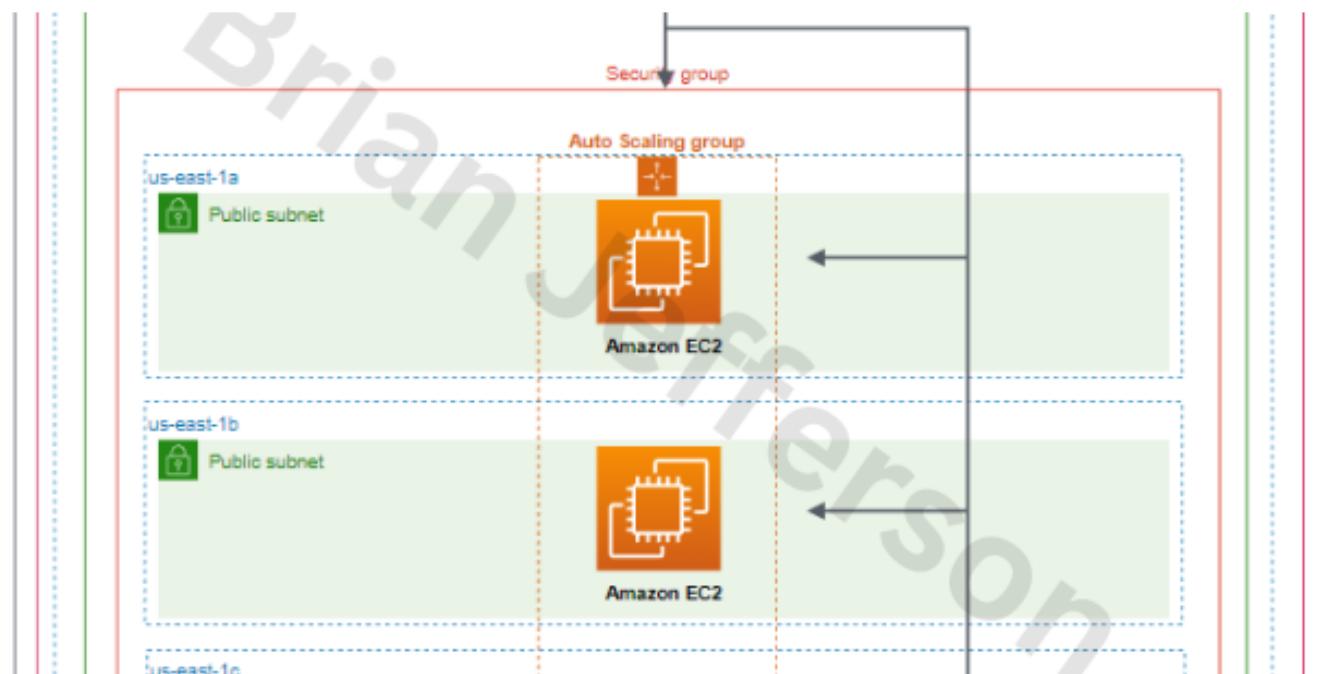
headintheclouds in AWS Tip

## Deploy WordPress and MySQL with Terraform on AWS EC2 Like a Pro

Prerequisites:

◆ · 14 min read · Jan 3

16 2



Brian Jefferson

# Ensuring High Availability: A Step-by-Step Guide to Creating an Auto-Scaling Group of EC2 Instances...

Scenario: We have been tasked to deploy a highly available web application on AWS by:

• 6 min read • May 1



## Lists



### Staff Picks

339 stories • 94 saves



### Stories to Help You Level-Up at Work

19 stories • 73 saves



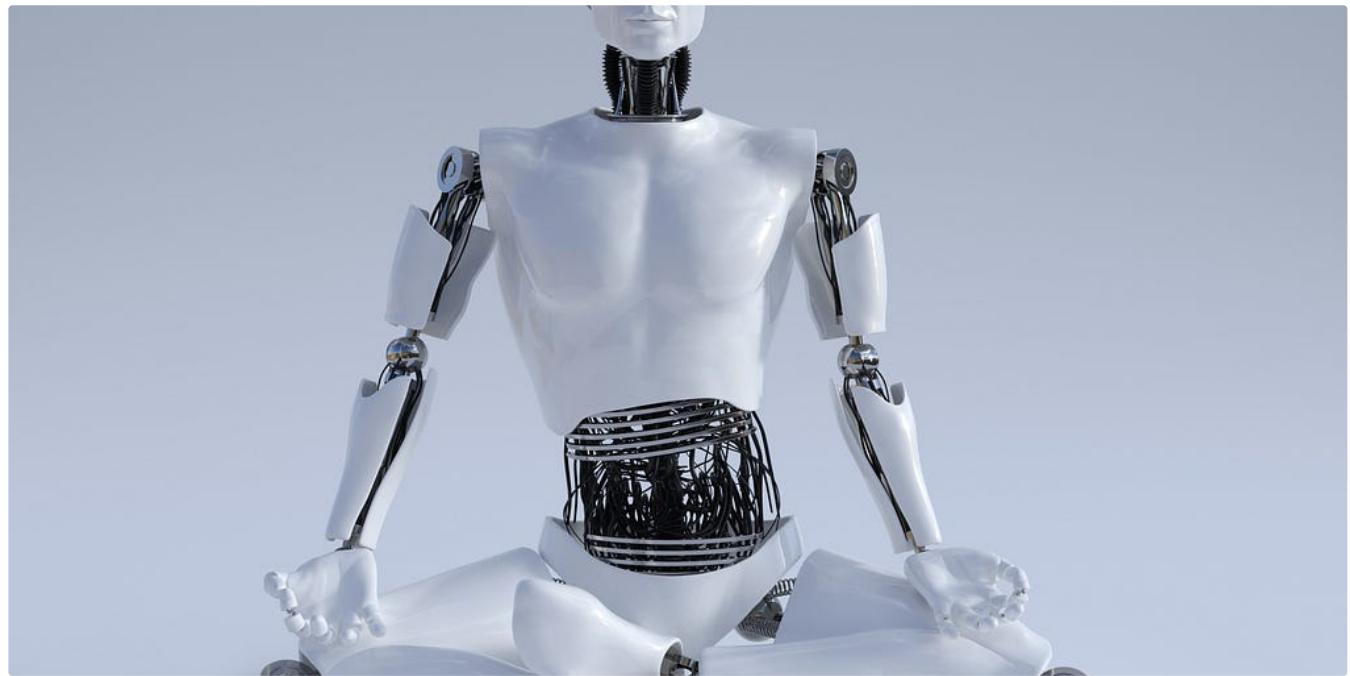
### Self-Improvement 101

20 stories • 130 saves



### Productivity 101

20 stories • 142 saves



The PyCoach in Artificial Corner

## You're Using ChatGPT Wrong! Here's How to Be Ahead of 99% of ChatGPT Users

Master ChatGPT by learning prompt engineering.

★ · 7 min read · Mar 17

 22K  379 Jack Roper in CodeX

## Terraform console command

Ever needed a quick and easy way to experiment with Terraform functions and expressions? In this short article, we will explore the...

★ · 4 min read · Mar 20

 70 



 Vickie Li

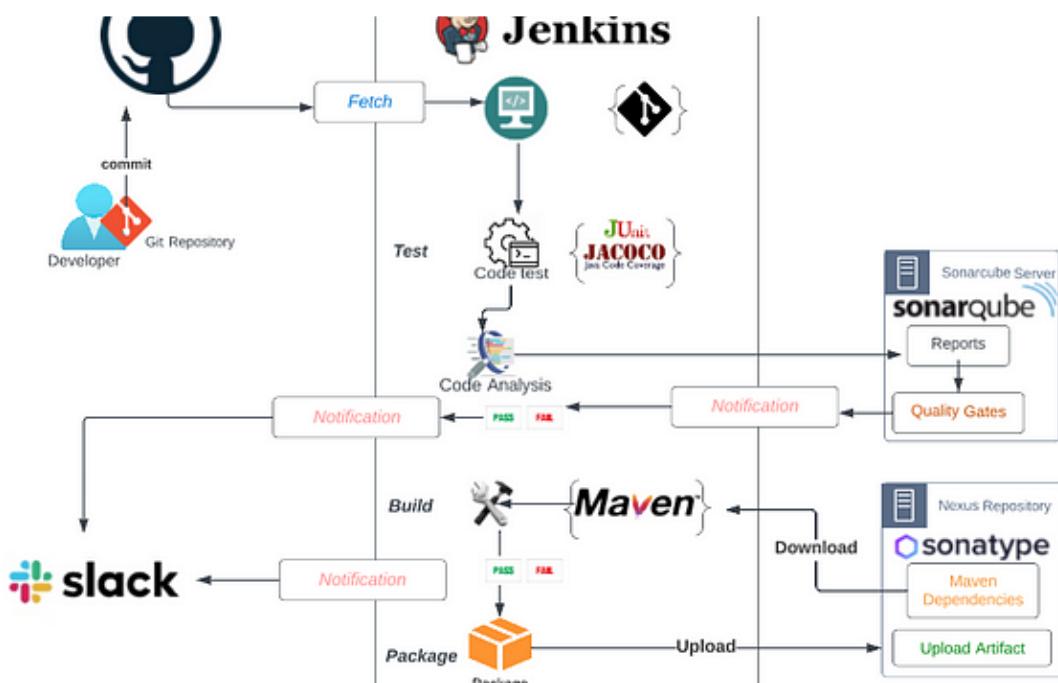
## Hacking LLMs with prompt injections

And ways hackers can attack GPT-based applications

◆ · 8 min read · 4 days ago

 345  2





Adewopo Olalekan Amos

## DevOps Project — CICD — Continous Integration of Java Web Application

## Project Source: DevOps Project by Imran Teli

14 min read · May 11



See more recommendations