M C A
Advanced Web Application Development

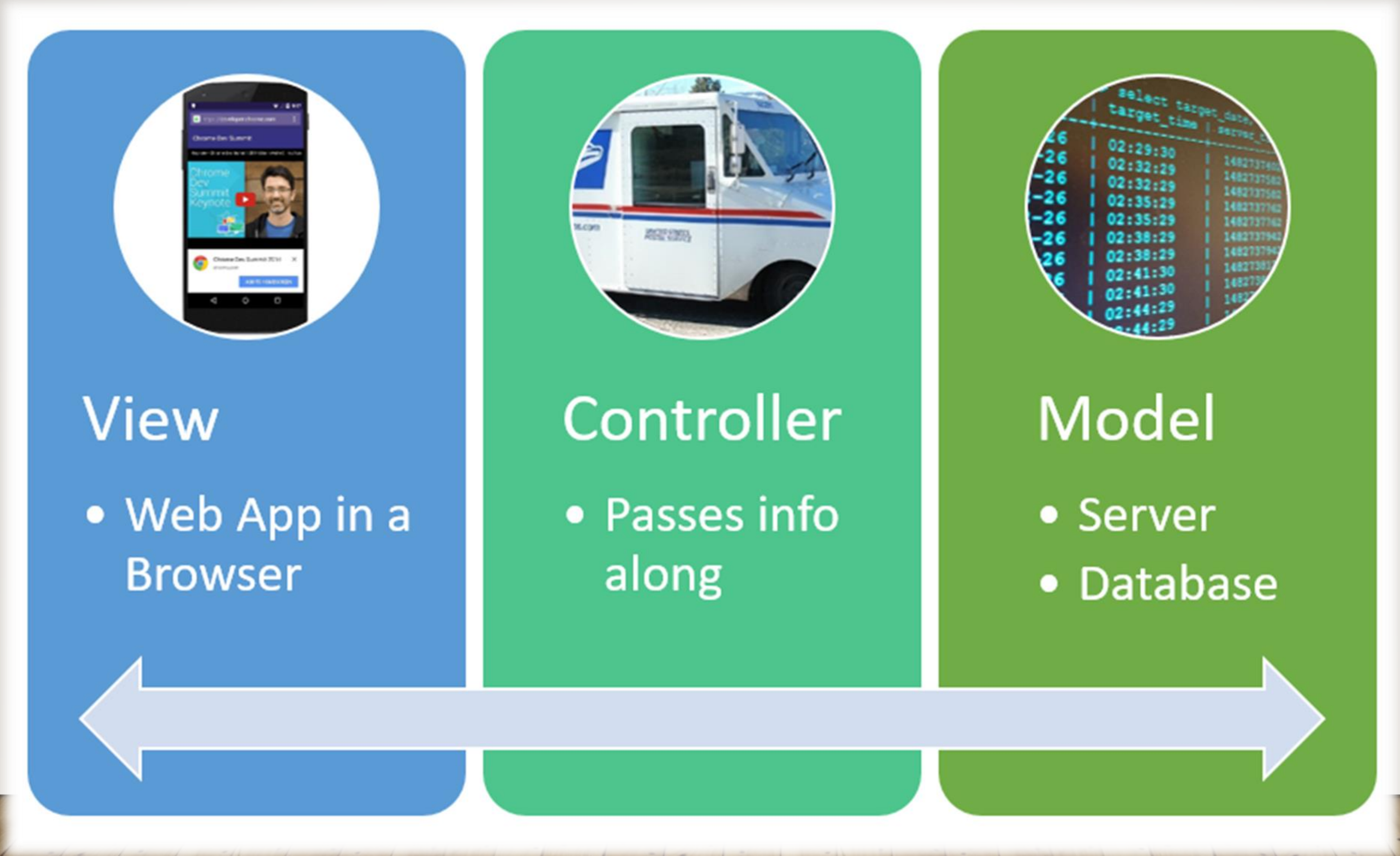
Dr. G.Babu

MODEL-VIEW-CONTROLLER

Model-View-Controller (MVC) is a software architectural pattern for implementing user interfaces.

It divides a given software application into three interconnected parts.

Model-View-Controller



View

- Web App in a Browser

Controller

- Passes info along

Model

- Server
- Database

MVC Example

View: User Interface

Controller: Engine

Model: Fuel

CONTROLLER

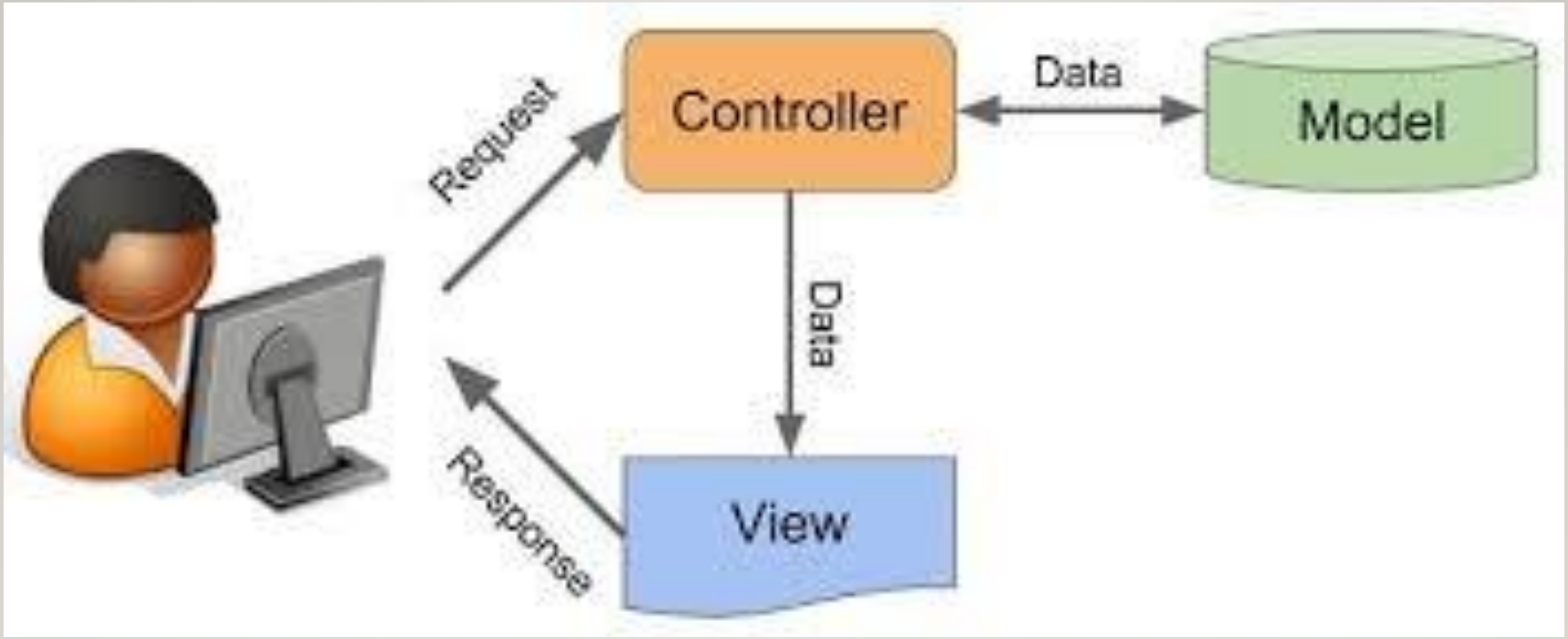
Car driving mechanism is another example of the MVC model.

Every car consist of three main parts.

View= User interface : (Gear lever, panels, steering wheel, brake, etc.)

Controller- Mechanism (Engine)

Model- Storage (Petrol or Diesel tank)



GETTING DATA FROM VIEW TO CONTROLLER IN MVC

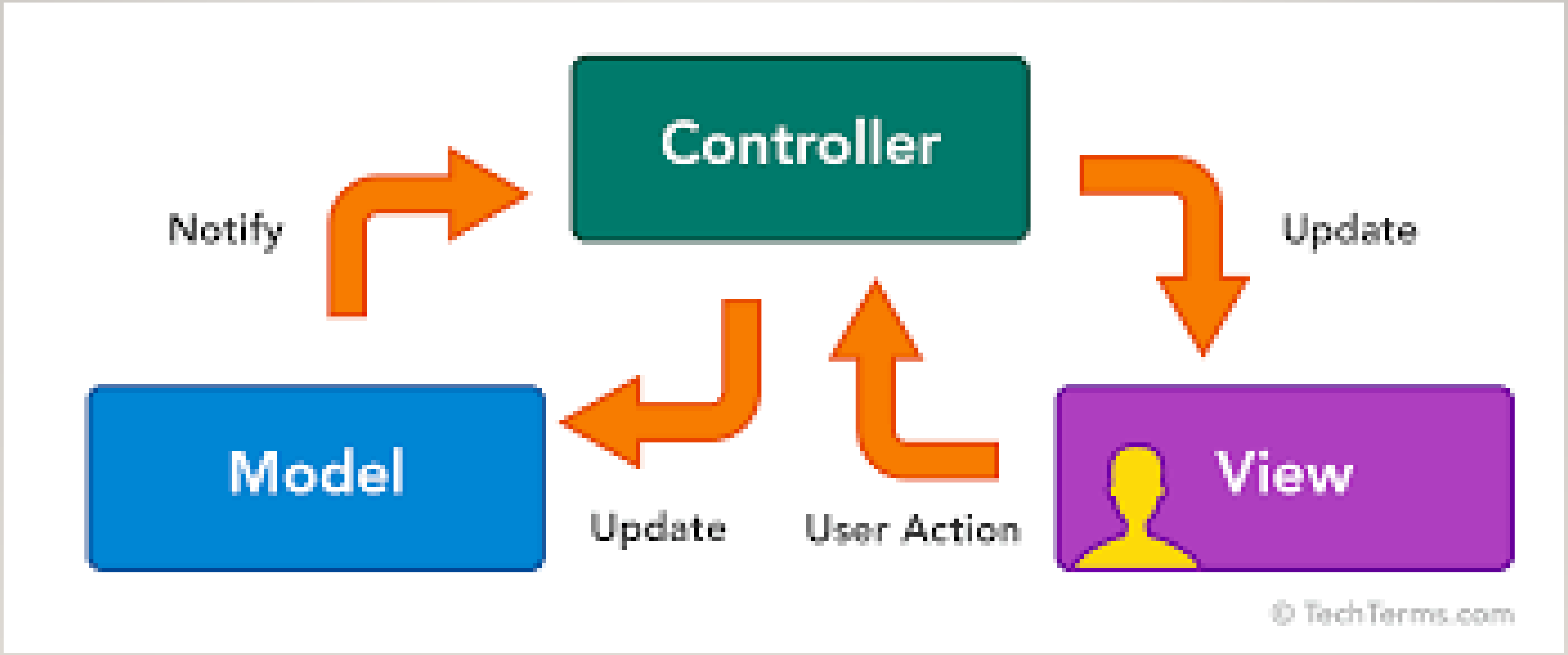
OBJECTIVES

Create Modular routes using Express

Describe Controllers and Views

Move data from view to controller and return it to the user to View in the browser.

BUILDING BASIC CONTROLLERS IN MVC



USES OF CONTROLLER

The Controller is that part of the application that handles the user interaction.

The controller interprets the mouse and keyboard inputs from the user, informing model and the view to change as appropriate.

USES OF CONTROLLER

A Controller send’s commands to the model to update its state (E.g., Saving a specific document).

The controller also sends commands to its associated view to change the view’s presentation

(E.g., scrolling a particular document).

CONTROLLER

Controller

The Controller is that part of the application that handles the user interaction. The controller interprets the mouse and keyboard inputs from the user, informing model and the view to change as appropriate.

EXAMPLE

Controllers act like intermediates between models and views, which are responsible for updating the model when the user manipulates the view.

In the above example of our photo gallery application, a controller would be responsible for handling changes the user made to the edit view for a particular photo, updating a specific photo model when a user has finished editing.

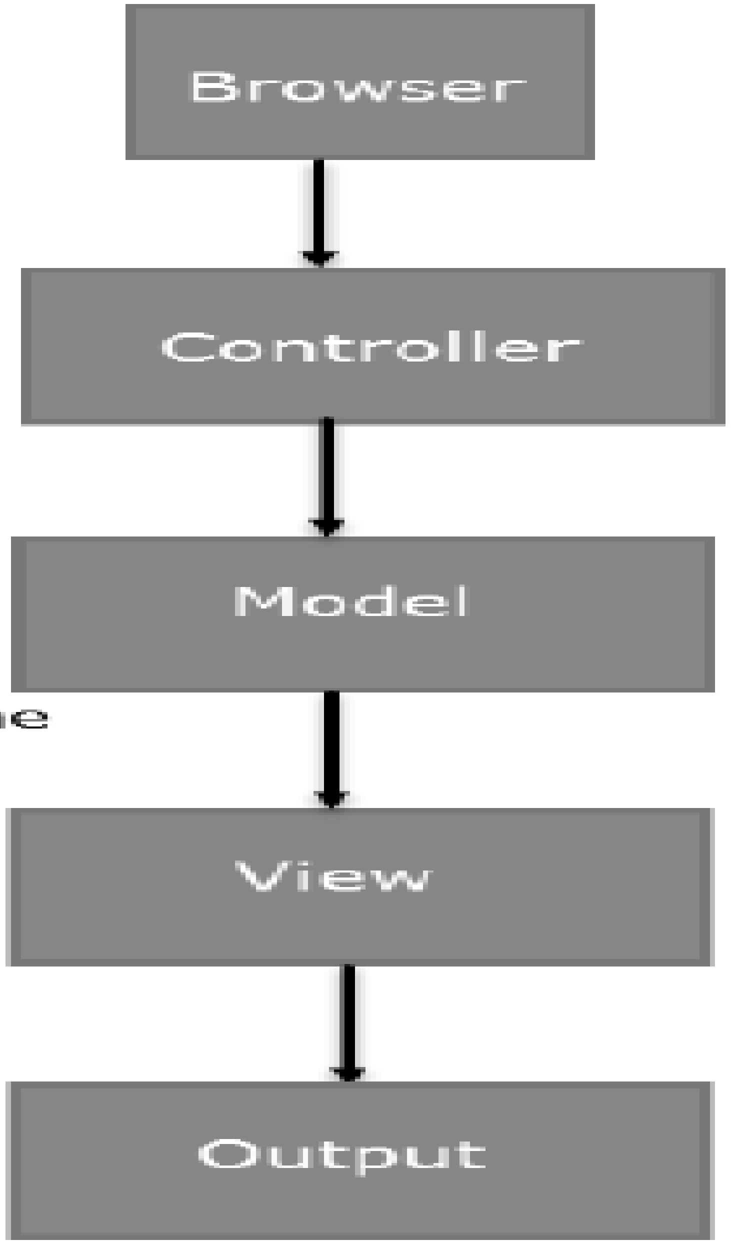
Browser sends request to the MVC Application

Incoming request directed to controller

Controller processes request and forms a data model

This model is passed to the appropriate View

The View renders the output



EXAMPLE

```
var PhotosController = Spine.Controller.sub({
init: function () {
this.item.bind( "update" , this.proxy( this.render ));
this.item.bind( "destroy", this.proxy( this.remove ));
},
render: function ()
{// Handle templating
this.replace( $( "#photoTemplate" ).tmpl( this.item ) );
Return  },
```

ACCESS DATA FROM A VIEW TO THE CONTROLLER

There are different ways used to access data from a view to the controller's action method.

Two types of methods to handle our browser request

1. **HTTP GET**
2. **HTTP POST.**

ACCESS DATA IN MVC

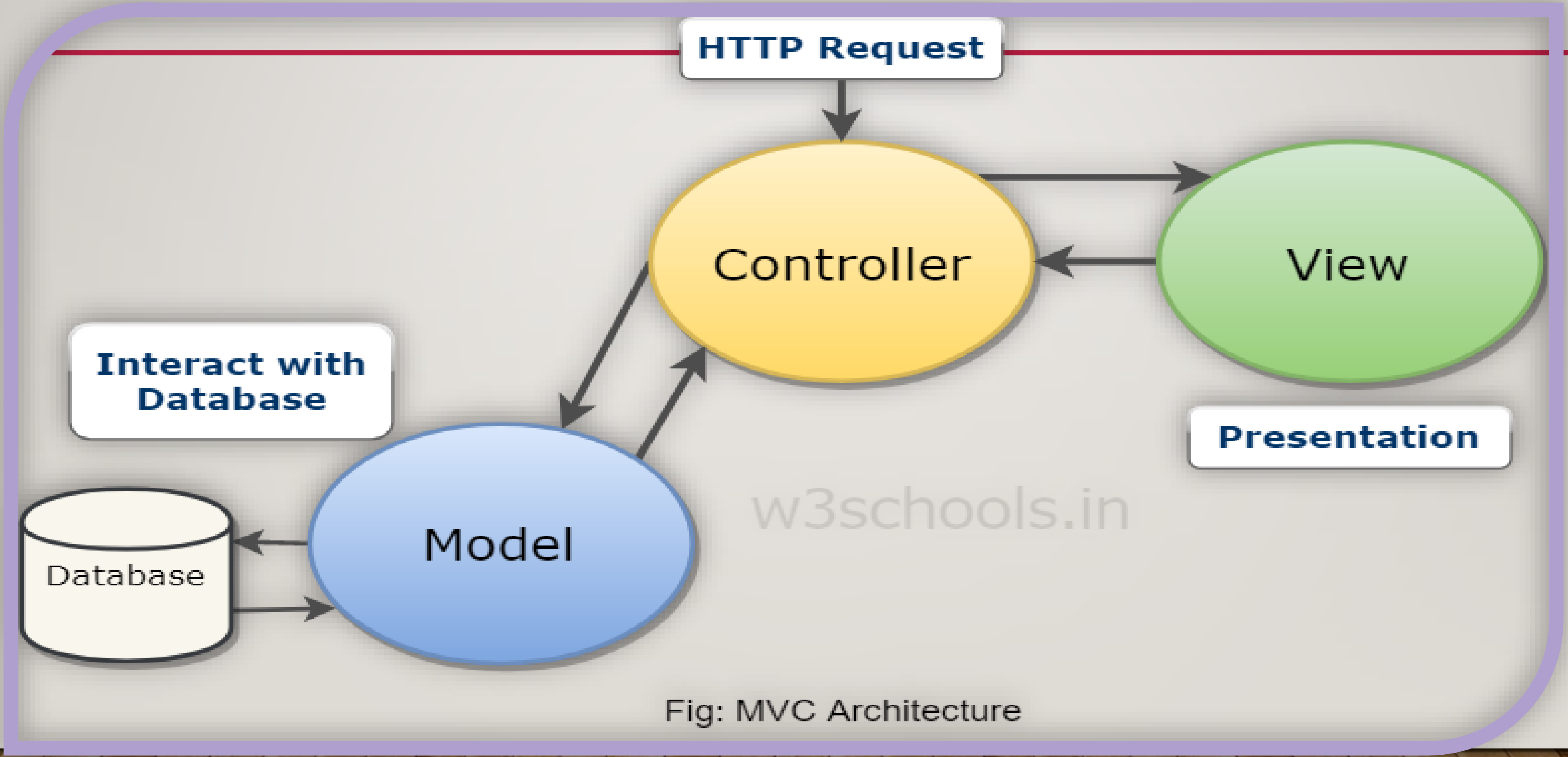


Fig: MVC Architecture

PASS DATA FROM CONTROLLER

1. Request comes in to the Controller
2. Controller makes a request to the model
3. Model returns a response to the controller
4. The controller (normally) performs some operations using the response from the model
5. The controller sends a response to the view
6. The view renders the response

THE OBJECTS OR COLLECTIONS OF OBJECTS

1. **Server** – to listen to and respond to HTTP requests
2. **Router** – to send the incoming requests to the correct controller
3. **Controllers** – to perform operations & interrogate the data
4. **Model** – to provide the data
5. **Views** – to provide the HTML rendering we're going to see in the browser

PASS DATA FROM CONTROLLER

Create a file called server.js with the following content:

```
var http_IP = '127.0.0.1’;

var http_port = 8899;

var http = require('http’);

var server = http.createServer(function(req, res)

{ require('./router').get(req, res); });

// end server() server.listen(http_port, http_IP);

 console.log('listening to http://' + http_IP + ':' + http_port);
```

GETTING DATA FROM THE MODEL

Using a view template to create View Bag

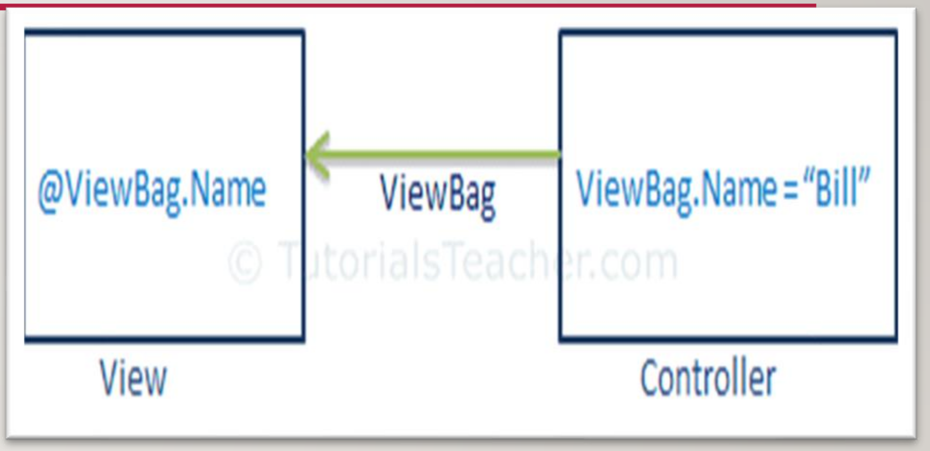
It is used to transfer temporary data from the controller to the view.

ViewBag only transfers data from controller to view

ViewBag is created on Server Side of the Web application and hence it is not possible to directly set it on Client Side using JavaScript.

VIEWBAG

The following figure illustrates the ViewBag.



I attaches Name property to ViewBag with the dot notation and assigns a string value "Bill" to it in the controller.

This can be accessed in the view like @ViewBag.Name.

MCQs

MCQS

MVC stand for

A. Model-View-Controller.

B.Modern-View-Controller.

C.Model-View-Constant.

D.None of these

MCQS

MVC stand for

A. Model-View-Controller.

B.Modern-View-Controller.

C.Model-View-Constant.

D.None of these

MCQS

MVC comes from which year?

A. 2003

B.2002

C.2004

D.2006

MCQS

MVC comes from which year?

A. 2003

B.2002

C.2004

D.2006

MCQS

1……helps you to maintain data when you move from controller to view.

A. View Bag
B. View Data
C. Temp Data
D. None of above

MCQS

1……helps you to maintain data when you move from controller to view.

A. View Bag

B. View Data

C. Temp Data

D. None of above

MCQS

2.What is true about view bag in MVC?

A. It is used to transfer temporary data from the controller to the view.

B.ViewBag only transfers data from controller to view

C.ViewBag values will be null if redirection occurs.

D.All of the above

MCQS

2.What is true about viewbag in mvc?

A. It is used to transfer temporary data from the controller to the view.

B.ViewBag only transfers data from controller to view

C.ViewBag values will be null if redirection occurs.

D.**All of the above**

WEB APPLICATION LIVE WITH HEROKU

WEB APPLICATION LIVE WITH HEROKU

HEROKU

OBJECTIVES

Developers use Heroku to deploy, manage, and scale modern web service application.

Web application framework used to create different front-end frameworks without infrastructure knowledge.

*

SaaS Enablement	Marketplace Custom Packaging Premium CDN & DNS Built-In Billing	Cloud Scripting WHMCS verizon CloudBlue
PaaS Management	App Deployment Auto-Scaling & Clustering CI/CD Automation Container Orchestration	GitHub GitLab kubernetes docker
IaaS Optimization	Containers Virtual Machines Network Storage	Virtuozzo KVM OPEN CONTAINER INITIATIVE ceph openstack

DEFINITION

Heroku is a cloud service provider and software development platform which facilitates fast and effective building, deploying and scaling of web applications

Fastest way to go from idea to URL, bypassing all infrastructure headaches.

GETTING STATED WITH HEROKU

Getting Started on Heroku

Step-by-step guides for deploying your first app and mastering the basics of Heroku

Heroku can deploy, run and manage applications written in Ruby, Node.js, Java, Python, Clojure, Scala, Go and PHP

DIFFERS FROM TRADITIONAL SERVER-BASED HOSTING

Heroku is an embodiment of web application development principles and differs from traditional server-based hosting in the following ways

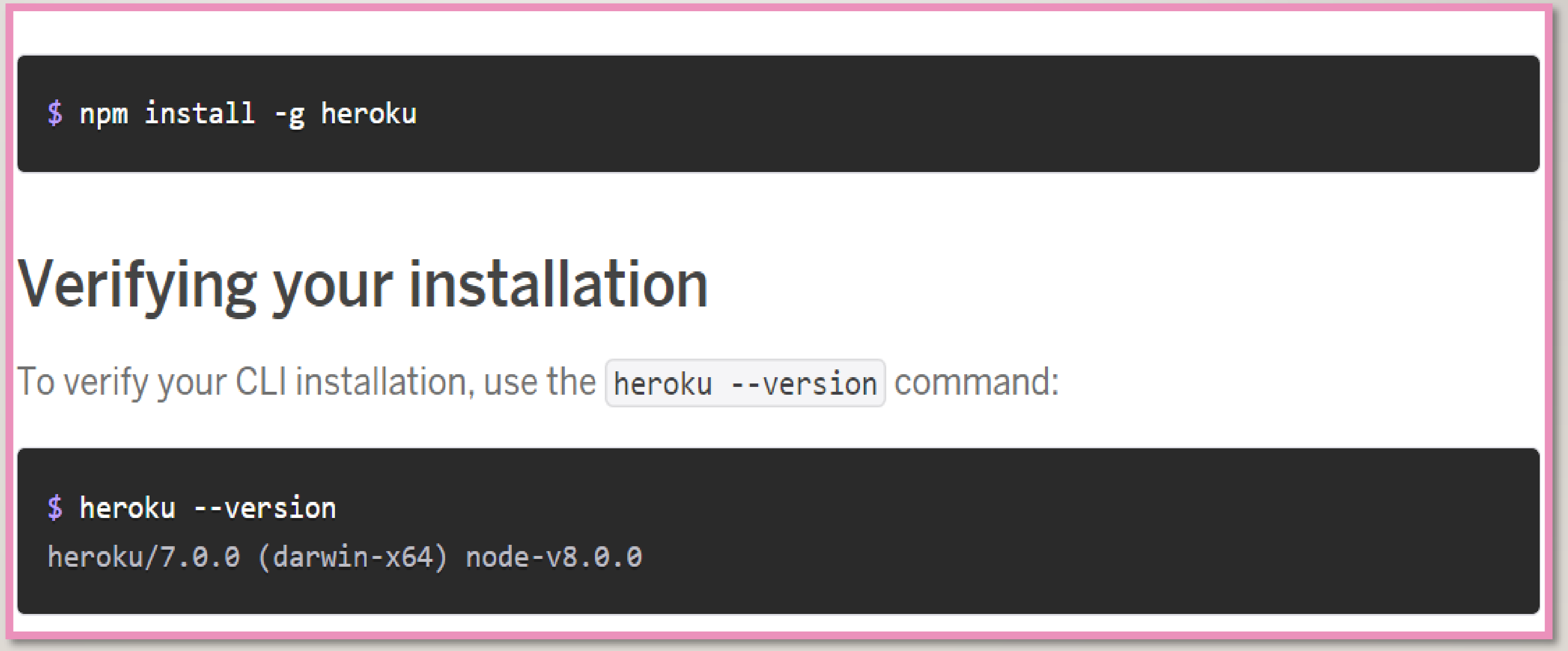
- It is application, not infrastructure, focused.
- It is a dynamic and distributed runtime environment.
- It utilizes a process-based execution model.

```
np
 npm install -g heroku
```

INSTALL HEROKU

Download and install the Heroku Command Line Interface (CLI).

•



```
$ npm install -g heroku
```

Verifying your installation

To verify your CLI installation, use the `heroku --version` command:

```
$ heroku --version
heroku/7.0.0 (darwin-x64) node-v8.0.0
```

DEPLOYING APPLICATIONS

An application is a collection of source code written in one of the languages, perhaps a framework, and some dependency description that instructs a build system as to which additional dependencies are needed in order to build and run the application.

Git is a powerful, distributed version control system that many developers use to manage and version source code.

When you create an application on Heroku, it associates a new Git remote

The Heroku platform uses Git as the primary means for deploying applications

$ git push heroku master

GITHUB

GITHUB is a Git repository hosting service has many of its own features.

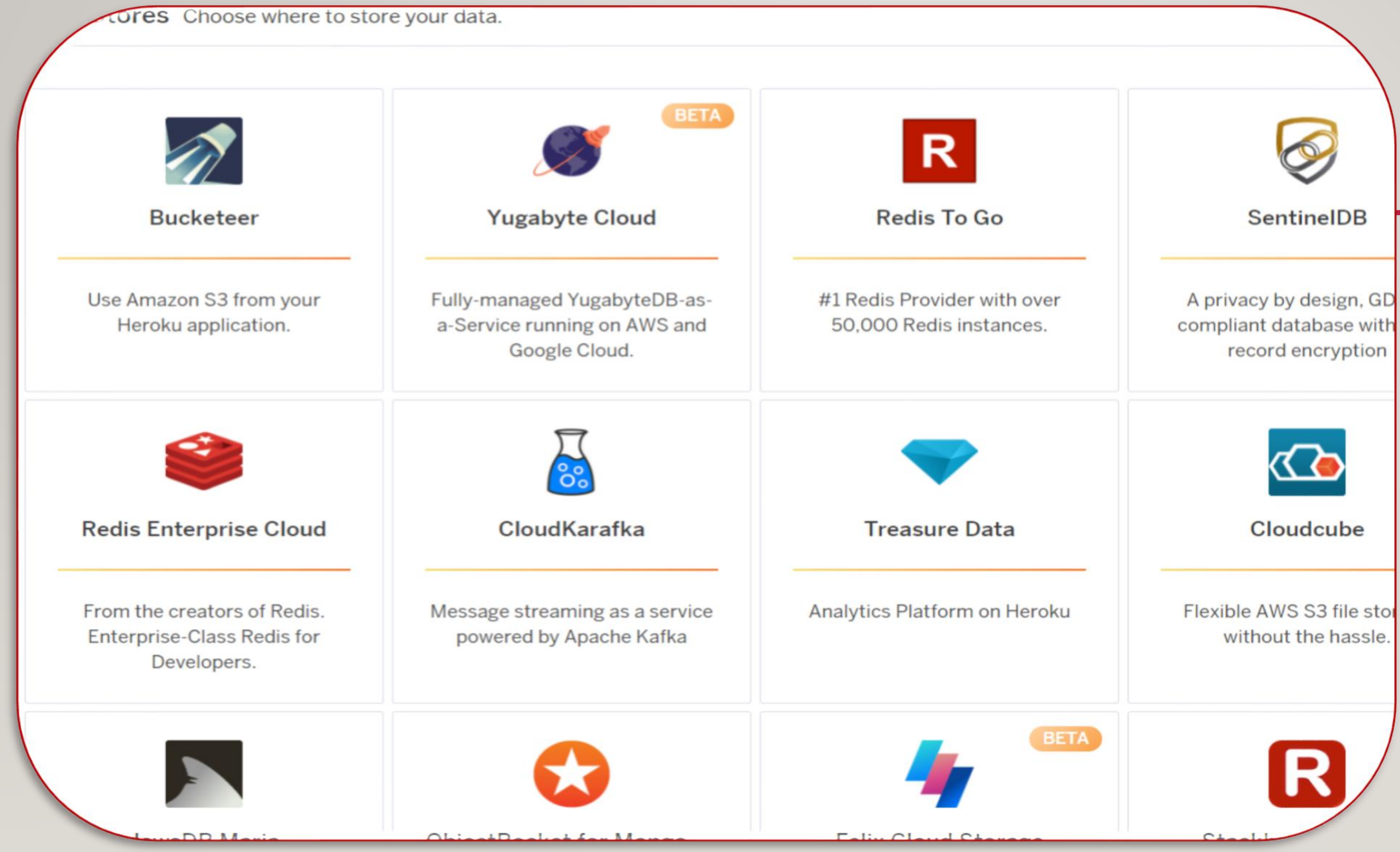
Git is a command line tool, GitHub provides a Web-based graphical interface.

ADD-ONS

Applications typically make use of **add-ons** to provide backing services such as databases, queueing & caching systems, storage, email services and more.

```
$ heroku addons:create heroku-redis:hobby-dev
```

DATA STORE



tores Choose where to store your data.

BETA

Bucketeer

Use Amazon S3 from your Heroku application.

Yugabyte Cloud

Fully-managed YugabyteDB-as-a-Service running on AWS and Google Cloud.

Redis To Go

#1 Redis Provider with over 50,000 Redis instances.

SentinelDB

A privacy by design, GD compliant database with record encryption

Redis Enterprise Cloud

From the creators of Redis. Enterprise-Class Redis for Developers.

CloudKarafka

Message streaming as a service powered by Apache Kafka

Treasure Data

Analytics Platform on Heroku

Cloudcube

Flexible AWS S3 file sto without the hassle.

BETA

RELEASES OF APPLICATION

Use the heroku releases command to see the audit trail of release deploys:

```
$ heroku releases
```

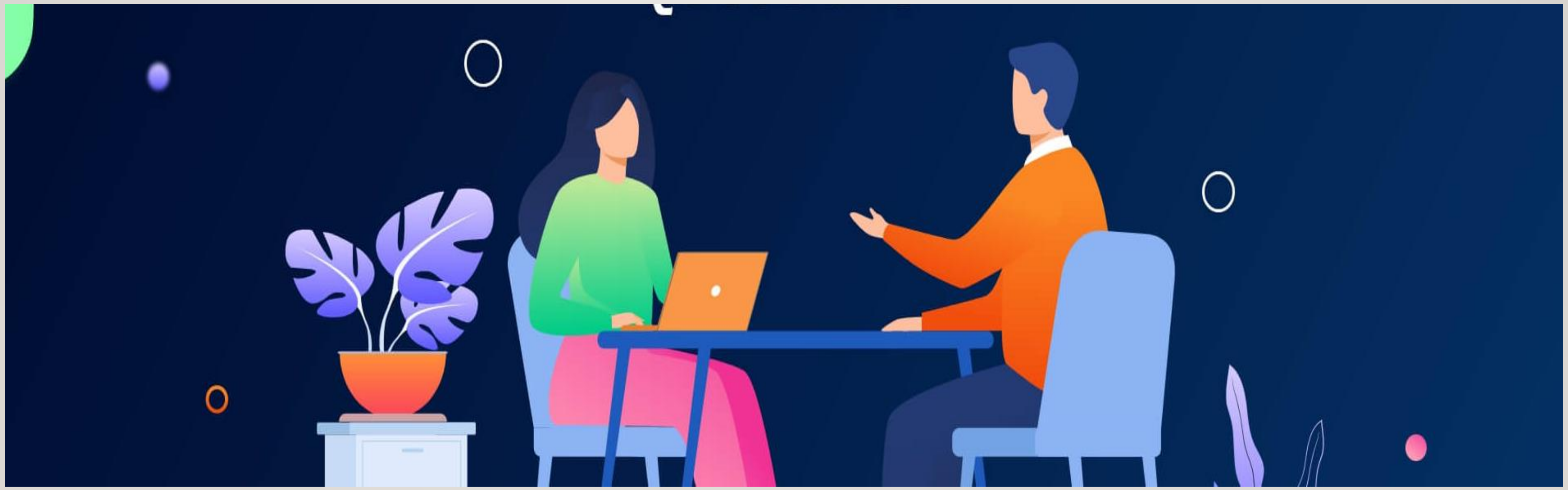
Every time you deploy a new version of an application, a new slug is created and release is generated.

As Heroku contains a store of the previous releases of the application, it's very easy to rollback and deploy a previous release

```
$ heroku releases:rollback v102
```

```
"dev": "nodemon ./bin/www"
```

MCQs



MCQS

HEROKU is _______________

1. Cloud platform as a service(PaaS).
2. Framework
3. Web Based Server
4. None of above

MCQS

HEROKU is _______________

1. Cloud platform as a service(PaaS).
2. Framework
3. Web Based Server
4. None of above

MCQS

GIT comes from

1.2005

2.2009

3,2004

4.2001

MCQS

GIT comes from

1.2005

2.2009

3,2004

4.2001

COURSE LEARNING OUTCOMES

- Explain importance of EUROKU
- Create Application Using GIT
- Describe ADD ON features
- Release the Application

Thank You