Silicon Valley Codecamp 2017

Extra Credit Assignment

Name: Lakshmanan Vivek

SJSU Student ID: 011794653

Email Address: viveklakshmanan@live.com

SVCC Code Camp Check-In badge

Title of the talk: **Getting Deep into Machine Learning with TensorFlow**

URL: https://www.siliconvalley-codecamp.com/Session/2017/getting-deep-into-machine-learning-with-tensorflow

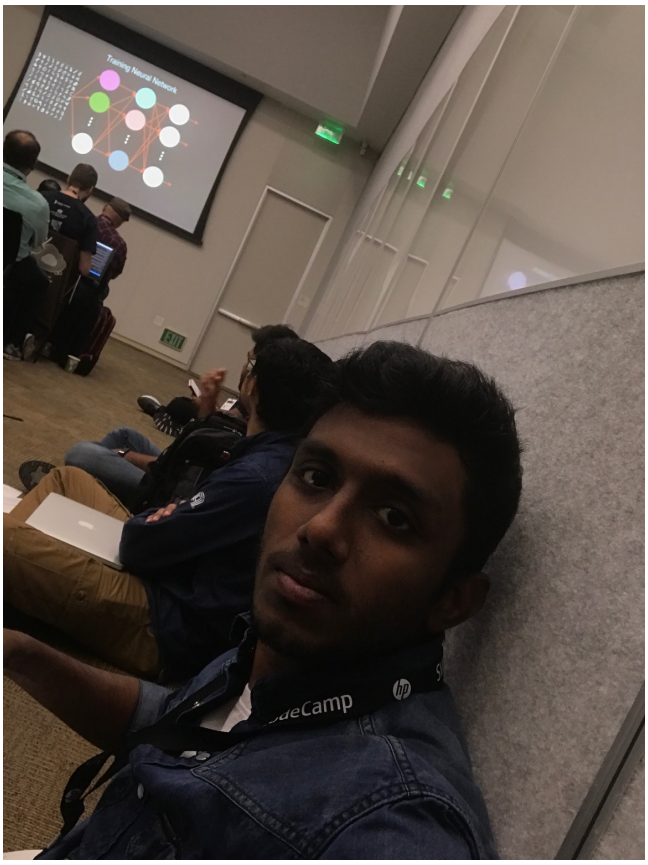
Opinions/Comments:

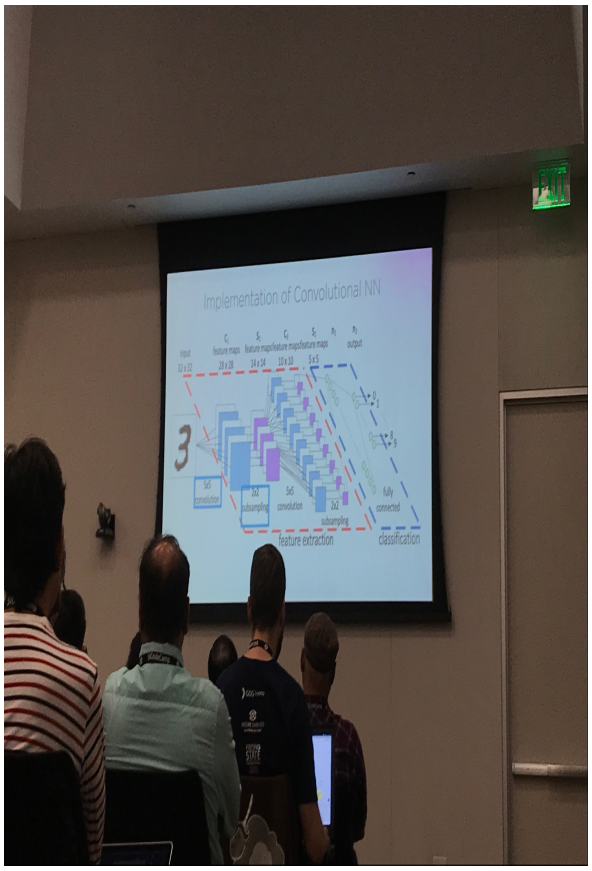
In this talk he has discussed about Deep Learning. He has first introduced us to the world of the deep learning by giving few real-time applications like hand-writing recognition, automatic image tagging, self-driving cars etc.

For hands-on he has shown us implementation of the Google's popular machine learning framework TensorFlow. He also spoke about how TensorFlow and keras work efficiently in implementing world-class AI solutions.

Overall this session was very informative as he has introduced me to the world of Deep Learning and TensorFlow, which is very new for me.

Pictures at the talk:





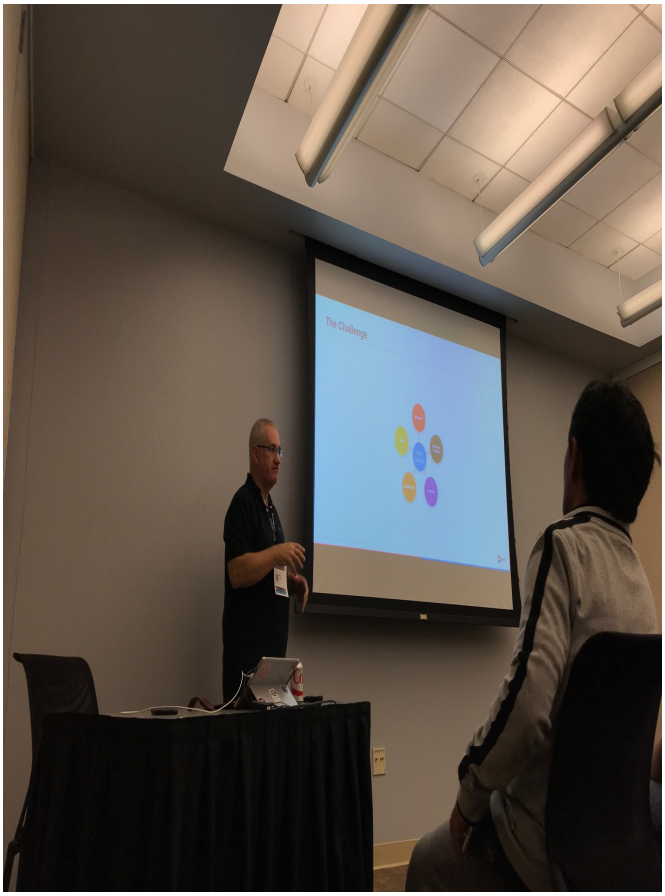
Talk: **Goodbye VMs - Hello Docker**

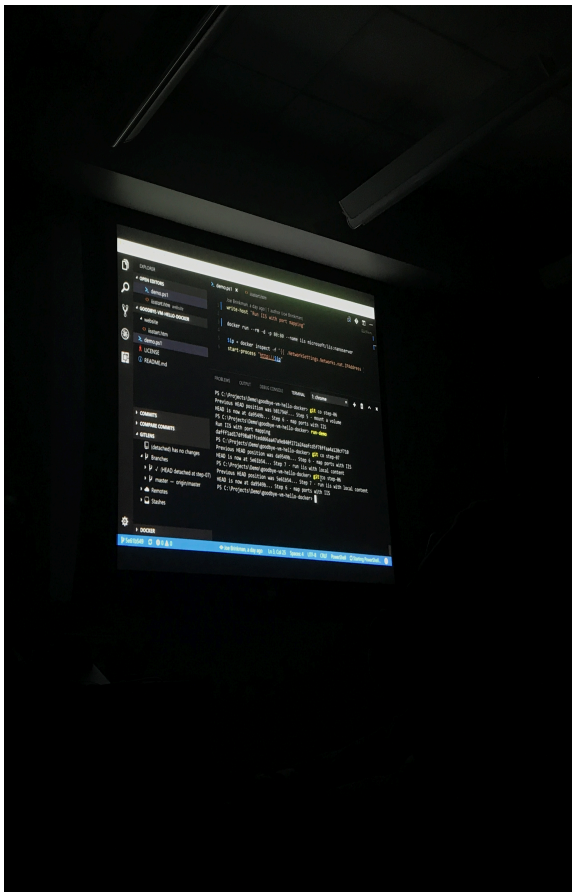
URL: https://www.siliconvalley-codecamp.com/Session/2017/goodbye-vms---hello-docker

Opinions/ Comments:
He spoke about the evolution of the virtual machines first and why we use it in the first place. Then he introduced the concept of dockers and containers. He explained why dockers are a lightweight solution for providing isolated environments and way better than virtual machines.

On the whole, in this presentation the speaker gave a very clear idea of how docker and container works. For the lab he has demonstrated how to create a docker and migrate an existing web application to the docker.

Pictures at the talk:





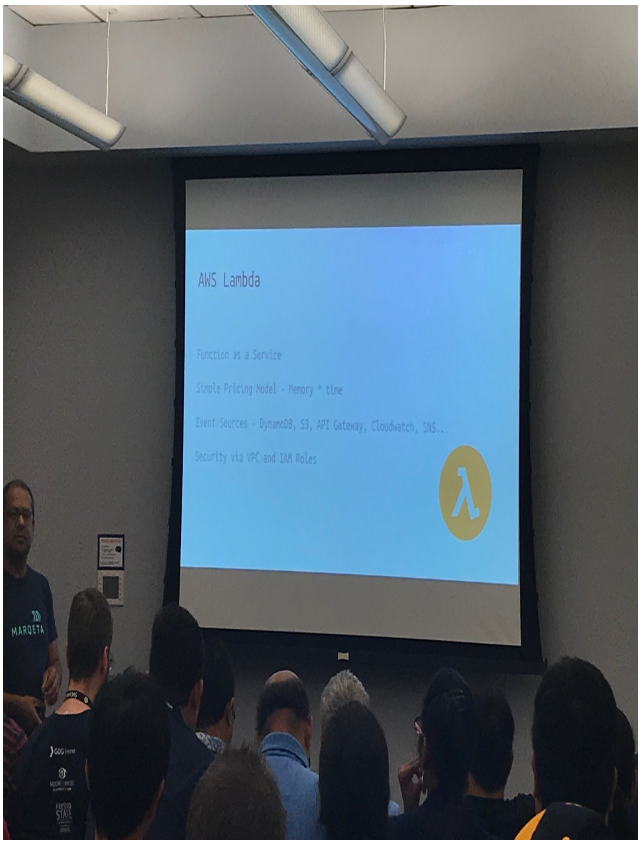
Title of the Talk: **AWS Lambda with Serverless Framework and Java**

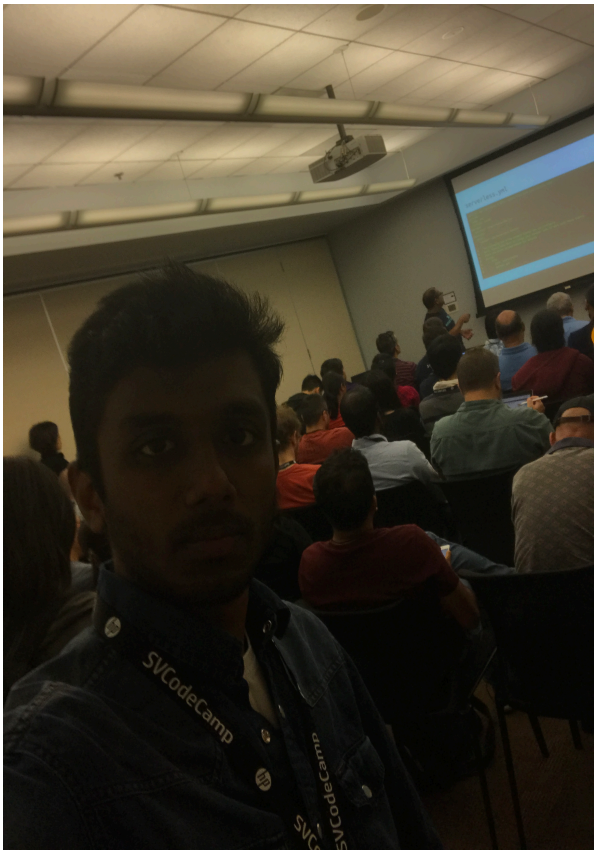
URL: https://www.siliconvalley-codecamp.com/Session/2017/aws-lambda-with-serverless-framework-and-java

Opinions/Comments:

In this presentation he spoke about the AWS Lambda with serverless framework and java. He explained the concept behind the AWS Lambda and why it is so powerful. For the lab he created and deployed a java-maven based AWS Lambda API. He showed us the command line interface to manage lambda and create serverless applications.

Pictures:





Title of the Talk: **Getting Started with The Go Programming Language (golang)**

URL: https://www.siliconvalley-codecamp.com/Session/2017/getting-started-with-the-go-programming-language-golang

In this presentation, the presenter gave a brief introduction of the go programming language and tells why it is superior to the other existing languages. He has discussed about the closure and concurrency in go lang too.

Extended Code/Project Assignment

Exercise 1: Understanding the data-types in go lang:

Here, in the program we are creating different variables with different data types to see how it works in go lang.

data-types.go

```
package main

import "fmt"

func main() {

	a := 24
	b := "golang"
	c := 0.65
	d := true
	e := "Hello, world!!!"
	f := `This is vivek`
	g := 'M'

	fmt.Printf("%v \n", a)
	fmt.Printf("%v \n", b)
	fmt.Printf("%v \n", c)
	fmt.Printf("%v \n", d)
	fmt.Printf("%v \n", e)
	fmt.Printf("%v \n", f)
	fmt.Printf("%v \n", g)
}
```

Exercise 2: Getting user input and printing it

In this program, we are getting the input value from user and printing on the console.

std_input.go

```
package main

import "fmt"


func main() {
	var name string
	fmt.Print("Enter your name: ")
	fmt.Scan(&name)
	fmt.Println("Hey there!", name)
}
```

Exercise 3: Closure in go lang

Go supports anonymous functions, which can form closures. Anonymous functions are useful when you want to define a function inline without having to name it.

This function addTo returns another function, which we define anonymously in the body of addTo. The returned function closes over the variable i to form a closure.

We call addTo, assigning the result to adder. This function value captures its own i value, which will be updated each time we call adder.

To confirm that the state is unique to that particular function, we have created and tested with a new adder variable.

```
package main

import "fmt"


func addTo() func() int {
   i := 0
   return func() int {
      i += 1
      return i
   }
}

func main() {

   adder := addTo()


   fmt.Println(adder())
   fmt.Println(adder())
   fmt.Println(adder())

   adder = addTo()
   fmt.Println(adder())
}
```

Exercise 4: Struts in go lang

Struts in go lang are similar to C language struts. Here in this program we create a strut with two strings and one int variable age. In the main function we create two person struct and assign values to them.

Struts.go:

```
package main

import "fmt"

type person struct {
	first string
	last  string
	age   int
}

func main() {
	p1 := person{"James", "Bond", 20}
	p2 := person{"Miss", "Moneypenny", 18}
	fmt.Println(p1.first, p1.last, p1.age)
	fmt.Println(p2.first, p2.last, p2.age)
}
```