Terraform

Terraform Class Outline

Day	Agenda
1	Terraform Introduction, first deployments
2	Architecture Deployments, modularizing your code
3	Managing state, working as a team, advanced language elements
4	Production grade deployments, Terraform testing

Prerequisites & Expectations

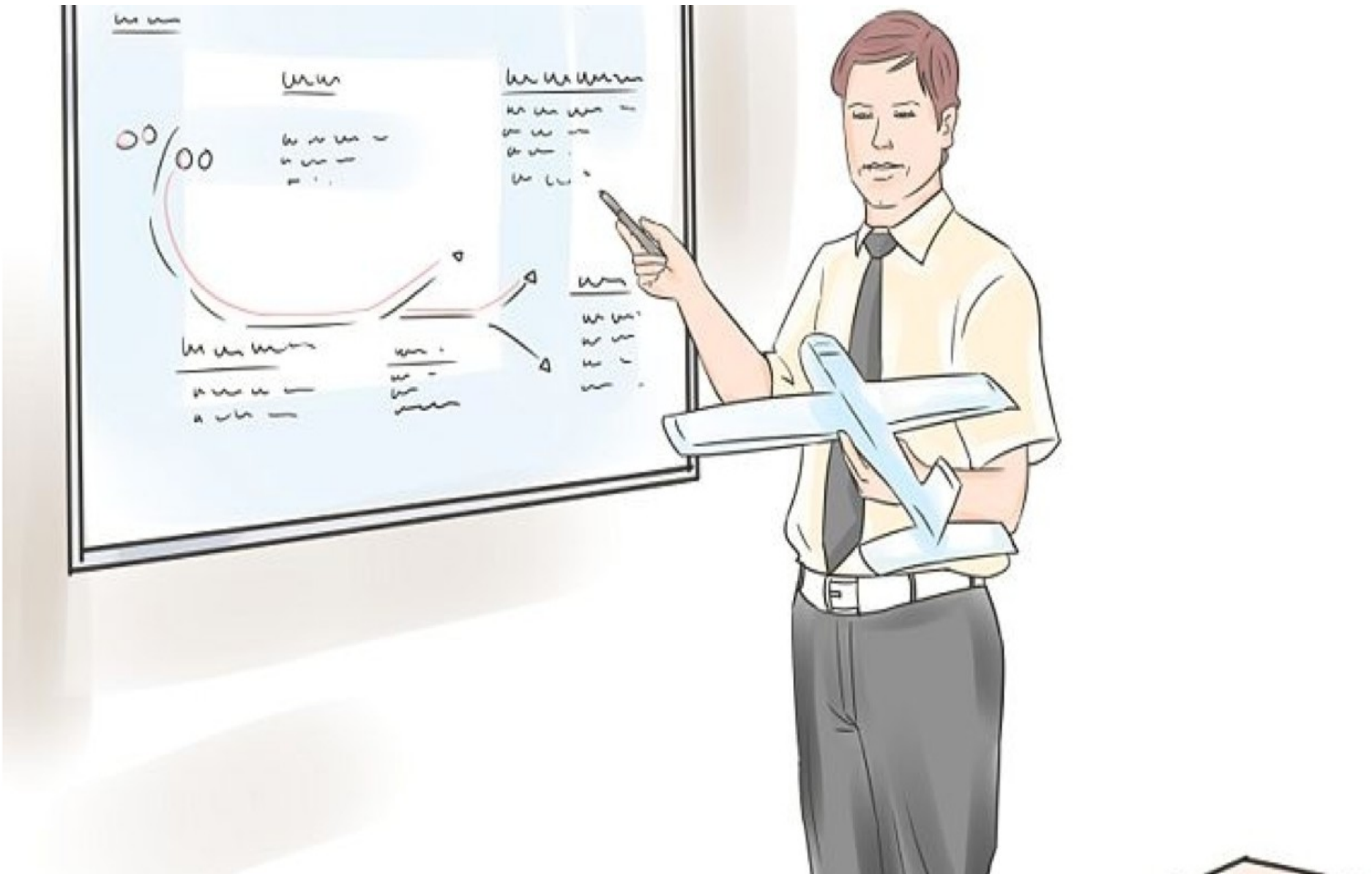
- Familiarity with a programming or scripting language
- Basic understanding of Linux development environment
  - Command line navigation
  - Editing files (e.g. using VI or nano)
- This is an **Infrastructure as Code (IaC) with Terraform** class
  - You may be missing some pre-requisites, that's OK
  - You should be willing to work hard, that is a must

Our Teaching Philosophy

- Emphasis on concepts & fundamentals
- No need to learn APIs by heart
- Highly interactive (questions, discussions, etc. are welcome)
- Hands-on (learn by doing)
- Build solid fundamentals of understanding and design

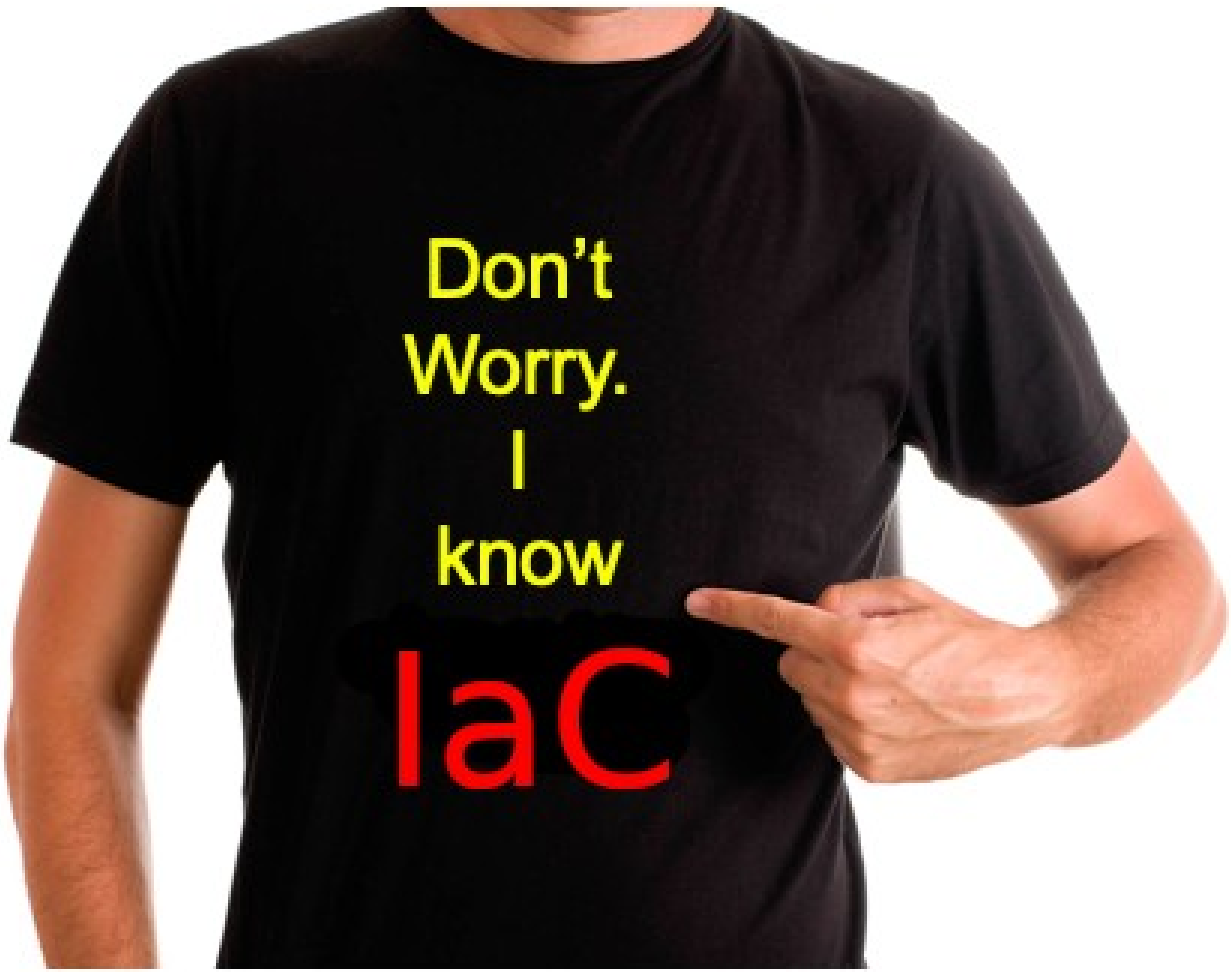
Analogy: Learning to Fly...

Instruction



+ Flight Time

After the Class...



About You and Me

- About Instructor
- About you
  - Your Name
  - Your background (developer, admin, manager, ...)
  - Technologies you are familiar with
  - Familiarity with Terraform or IaC (scale of 1 - 4 ; 1 - new, 4 - expert)
  - Something non-technical about you!(favorite ice cream flavor, hobby, etc.)
  - Something non-technical about you!(favorite ice cream flavor, hobby, etc.)

Class Logistics

- Instructor’s contact information
- Slides
  - For each session, slides will be emailed out or delivered via virtual classroom or provided in a repository
- Server nodes
  - Provided in the cloud
- Labs
  - Provided in the cloud
  - Will be delivered in a ZIP file for your future reference
  - Or be supplied in a repository

Typographic Conventions

- Software code in the text uses a fixed-width code font:
  - catalog: Catalog = new CatalogImpl
  - Code fragments are the same, e.g., catalog.speakTruth
  - We use **bold/color** text for emphasis
  - Filenames are in italics, e.g., `Catalog.scala`
  - Longer code examples appear in a separate code box:

```
1 object TestApp {  // Basic Spark App (Scala)
2   def main(args: Array[String]) {
3         val sc = new SparkContext(
4       new SparkConf().setMaster("local").setAppName(”TestApp")
5         val totalWords = sc.textFile(“file”)
6                 .flatMap(l => l.split(" ")).count()
7         println (“# lines : “ + totalWords)
8     }
9 }
```

Questions?

- Any questions?