DevOps Course Summary

Lesson 1

What is DevOps?

• First of all, DevOps is not a title/job/profession/set of tools or software. DevOps is a culture!

• A culture where Dev and Ops teammates cooperate.

• DevOps culture is talking about:

- Engineers empowerment - giving engineers more responsibility over the whole application lifecycle process. (dev -> test -> deploy -> monitor).
- Test-Driven Development - write tests before you write code. Unit tests, integration tests, and system tests. It will help increase the quality of your service and give you more confidence to release faster and more frequently.
- Automation - automate everything that can be automated - test automation, infrastructure automation, deployment automation, etc.
- Monitoring - monitor your apps, and build monitoring alerts well. It should save you time. Don't flood with metrics and alerts.
- Self-service - provide self-service for any framework you build or anything you do. Don't be a bottleneck.

Benefits

• Speed
  - Ensure faster time-to-market/delivery times that improve ROI.
  - DevOps is basically the application of Agile principles, and thus the result is faster software development, ensuring more frequent delivery.

• Reliability

  - Early detection and faster correction of defects that helps provide the best services and robust features that must be delivered to customers.

• Cultural
  - Happier, more productive teams.
  - Higher employee engagement.
  - Improved communication and collaboration.

Methodologies 2 Waterfall

• Gather and document requirements.
• Design.
• Code and unit test.
• Perform system testing.
• Perform user acceptance testing.
• Fix any issues.
• Deliver the finished product.
• The biggest problem with this approach is that the customer's needs usually change during the development phase, which causes delivering software that doesn't meet the revised needs, or you spend a lot of time and money changing plans midway.

Agile

• The idea in Agile is to develop software in small iterations (known as iterations) and be able to adapt to the changing customer needs better than in Waterfall.
• However, some of this model's disadvantages are:
  - Budget goals and deadlines are often missed.
  - New features break old functions.

Agile & DevOps

• DevOps brings more flexibility on top of the Agile model.
• With continuous integration (CI) and continuous delivery (CD) pipelines, you can make sure that you can release often and that the releases actually work and meet the customer's needs.
• Cooperation between the development team and IT operations ensures that also the used tools are streamlined and do not form bottlenecks.
• With effective tools, repetitive work can be automated, and transparency is improved.

CI (Continuous integration)

• Continuous Integration (CI) is a development practice where developers integrate code into a shared repository frequently, preferably several times a day.

• Each integration can then be verified by an automated build and automated tests.

• One of the key benefits of integrating regularly is that you can detect errors quickly and locate them more easily.
• As each change introduced is typically small, pinpointing the specific change that introduced a defect can be done quickly.
• In recent years, CI has become a best practice for software development and is guided by a set of key principles. Among them are revision control, build automation, and automated testing.
• Some popular tools used in CI, which will be discussed later, are:
    o Jenkins o Travis CI o Bamboo

CD (Continuous Delivery)

Continuous delivery is an extension of continuous integration to ensure that you can release new changes to your customers quickly and sustainably.

• This means that on top of having automated your testing, you also have automated your release process, and you can deploy your application at any point in time by clicking on a button.

• In theory, with continuous delivery, you can decide to release daily, weekly, or whatever suits your business requirements.

• However, if you truly want to get the benefits of continuous delivery, you should deploy to production as early as possible to make sure that you release small batches that are easy to troubleshoot in case of a problem.

CD (Continuous Deployment)

Continuous deployment goes one step further than continuous delivery.

• With this practice, every change that passes all stages of your production pipeline is released to your customers.

• There's no human intervention; only a failed test will prevent a new change from being deployed to production.

• Continuous deployment is an excellent way to accelerate the feedback loop with your customers and take pressure off the team, as there isn't a Release Day anymore.

• Developers can focus on building software and see their work go live minutes after they've finished working on it

Python

A clear and powerful object-oriented multi-purpose programming language. Some features of Python are:

a. Python supports object-oriented programming with classes and multiple inheritance.
b. Code can be grouped into modules and packages.
c. Python's automatic memory management frees you from manually allocating and freeing memory in your code.
d. Runs anywhere, including Mac OS X, Windows, Linux, and Unix, with unofficial builds also available for Android and iOS.
e. It is a free open-source language

Python Interpreter

- Python is an interpreted language which means that our code works as such:
  - Instructions (code) are read
  - Code evaluated
  - Result is returned for each instruction in a sequence.

- And it is the interpreter's job.

- Unlike compiled languages, in which the whole context is passed through a compiler, which in return gives a translational unit, the Standard interpreter for Python is written in C.

- There is also a version of Python written in Python, which instead of Python bytecode generates Python byte code (Jython).

- One of the advantages is the increased productivity it provides since there is no compilation step, unlike many other languages.

Data types

Data type is a representation of data within the code.

Type	Example
• Numeric: Integer, Float	x = 10 x = 1.0
• String	x= 'Mike'
• Boolean	y = True x = False
• List	my_list = [10, 20, 30]
• Tuple	my_tuple = ('Brett', 'Cisco', 'Cary',2015)
• Dictionary	my_dict = {"one":1, "two":2}

- Numbers
  - o int – whole number**: X = 2**
  - o float – a floating point number: **y = 2.2**
  - o long – are large whole numbers (happens automatically) when a number is big.

• Boolean
  - o Can hold one value out of the following two values: **True/False is Here = True.**

• String
  - o A string is a sequence of characters, which is marked in a single/double quote: **Name = "John"**

• Data structures –
  - o List, Tuple, Dictionary will be discussed later.

https://en.wikibooks.org/wiki/Python_Programming/Variables_and_Strings

Operators

• Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result.

• Operator Types:

a. Arithmetic operators.

b. Comparison operators.

c. In Python each **operator** should have space **before** and **after**.

Simple Assignment Operator

= Simple assignment operator

Arithmetic Operators

+ Additive operator (also used for String concatenation)
- Subtraction operator
* Multiplication operator
/ Division operator
% Remainder operator

Equality and Relational Operators

== Equal to
!= Not equal to
> Greater than
>= Greater than or equal to
< Less than
<= Less than or equal to

https://www.w3schools.com/python/python_operators.asp

Conditions

Condition is a Boolean statement which can generate true/false value.

https://docs.python.org/3.6/reference/compound_stmts.html

I - Simple if

```
a = 1
b = 2

If a > b:
        print("a is bigger")
if b > a:
        print("b is bigger")
if ..... :
```

Else will can be written right after the if statement.

```
a = 1
b = 2

if a > b:
        print("a is bigger")
else:
        print("b is bigger")
```

Elif is used to create conditions system.
• The elif statement allows you to check multiple expressions for TRUE and execute a block of code as soon as one of the conditions evaluates to TRUE.
• Similar to the else, the elif statement is optional.
• Unlike else, there can be a few of elif statements following an if.
• As soon as one condition is met, the rest will not be tested.

```
a = 1
b = 2

if a > b:
   print("a is bigger")
elif a == b:
   print("equals")
elif a != b:
   print("not equals")
else:
   print("none")
```

Indentation

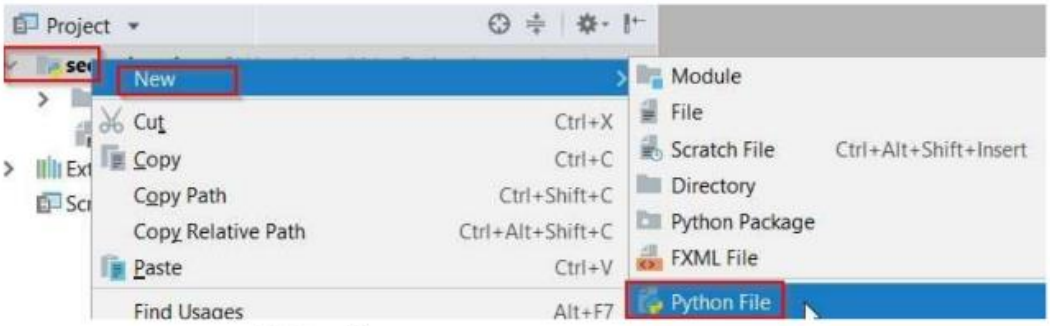
• Unlike many other languages, where each block of code is marked using brackets { } , in Python, each scope (block of code), is marked by using indentation.
• Indentation basically means spaces.
• Each Indentation should be multiple of four (4/8/12/...).
• At the below code, we have 4 spaces before print, to associate it with the if statement above.

```
If 1 > 0:
----print("a is bigger")
```

- **No spaces will cause an error.**

```
If 1 > 0:
print("a is bigger")
```

1. To create a new Python file: Right click on the project folder → New → Python file:



2. Type a name to your file (any)
3. Type: print ("your_name")
4. Right click → Run <your_file_name>
5. Let's start coding...