Bring your own serverless environment

Michael Pollett
Who are Datacamp
Why is this serverless
You’re already doing it
Building a business on serverless
Security detour
Application
What does DataCamp do?

1. **Learn**
   Acquire new skills. Choose from over 100 intuitive Courses on R, Python, SQL, Git, Shell, ...

2. **Practice**
   Sharpen and train your newly learned skills. Take bite-sized, fun Practice Challenges.

3. **Build**
   Apply your data science skills to real-world problems. Start hands-on data Projects.

- Online learning platform
- Multiple languages/tools
- Learn by doing
Hello Python!

Filip Schouwenaars
Learning Data Science

INTRO TO PYTHON FOR DATA SCIENCE

Got it!
What does DataCamp do?

Type conversion

Using the `+` operator to paste together two strings can be very useful in building custom messages.

Suppose, for example, that you’ve calculated the return of your investment and want to summarize the results in a string. Assuming the floats `savings` and `result` are defined, you can try something like this:

```python
savings = 108
result = 100 * 1.30 + 7

# Fix the printout
print("I started with $" + savings + " and now have $" + result + ". Awesome!")
```

However, as you cannot simply sum strings and floats, you’ll need to explicitly convert the types of your variables. More specifically, the `str()` function is needed to convert a value into a string, `str(savings)`.

Functions like `str()`, `float()`, and `bool()` will help you convert any type.

Run the code on the right. Try to understand the error message. Find the right such that the printout runs without errors. Use the `str()` function to convert the variables to strings.

Use `str()` to convert `pi_string` to a float and store this float as a new variable, `pi_float`.
Why is this serverless
Why is this serverless

The Python Interface

In the Python script on the right, you can type Python code to solve the exercises. If you hit Run Code or Submit Answer, your python script (script.py) is executed and the output is shown in the IPython Shell. Submit Answer checks whether your submission is correct and gives you feedback.

You can hit Run Code and Submit Answer as often as you want. If you’re stuck, you can click Get Hint, and ultimately Get Solution.

You can also use the IPython Shell interactively by simply typing commands and hitting Enter. When you work in the shell directly, your code will not be checked for correctness so it is a great way to experiment.
Why is this serverless

The Python Interface

In the Python script on the right, you can type Python code to solve the exercises. If you hit Run Code or Submit Answer, your python script (script.py) is executed and the output is shown in the IPython Shell. Submit Answer checks whether your submission is correct and gives you feedback.

You can hit Run Code and Submit Answer as often as you want. If you’re stuck, you can click Get Hint, and ultimately Get Solution.

You can also use the IPython Shell interactively by simply typing commands and hitting Enter. When you work in the shell directly, your code will not be checked for correctness so it is a great way to experiment.
Why is this serverless

**The Python Interface**

In the Python script on the right, you can type Python code to solve the exercises. If you hit Run Code or Submit Answer, your python script (script.py) is executed and the output is shown in the IPython Shell. Submit Answer checks whether your submission is correct and gives you feedback.

You can hit Run Code and Submit Answer as often as you want. If you’re stuck, you can click Get Hint, and ultimately Get Solution.

You can also use the IPython Shell interactively by simply typing commands and hitting Enter. When you work in the shell directly, your code will not be checked for correctness so it is a great way to experiment.
Inconspicuous serverless
version: 2.1

jobs:
  build:
    environment:
      TAG: snapshot
docker:
    - image: datacamp/docker-deploy
steps:
  - checkout
  - setup_remote_docker
  - deploy:
    name: Deploy
command:
docker build \
  -t ${ECR_URL}/s{PROJECT_REPO_NAME}:builder

package main

import {
  "fmt"
  "log"
  "path/filepath"
}

func main() {
  files, err := filepath.Glob("/etc/*")
  if err != nil {
    log.Fatal(err)
  }
  fmt.Println(files)
}

[/etc/group /etc/hosts /etc/passwd /etc/resolv.conf]
Building a business on serverless
We’ve all done ‘dockerising’

How about a whole business model based on serverless tech
Behind the scenes
Behind the scenes
Behind the scenes
A world without containers?
A world without containers

- Virtualization
- Emulation
- Unparsed code
- Multiple-choice ??
The downsides
The downsides

- Security...
- Code execution within our environment
- Docker as a sandbox
6.2 Test for dynamic scripting injection

<table>
<thead>
<tr>
<th>CVSS score</th>
<th>Risk</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.5</td>
<td>High</td>
</tr>
</tbody>
</table>

The vulnerable locations have been outlined below.

<table>
<thead>
<tr>
<th>Location</th>
<th>/input?sid=&lt;sid&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Method</td>
<td>POST</td>
</tr>
<tr>
<td>Params</td>
<td>command</td>
</tr>
<tr>
<td>Payload</td>
<td>As an example in a Python process:</td>
</tr>
<tr>
<td></td>
<td>import subprocess</td>
</tr>
<tr>
<td></td>
<td>cmd = &quot;uname -a&quot;</td>
</tr>
<tr>
<td></td>
<td>x = subprocess.Popen(cmd, shell=True, stdout=subprocess.PIPE, stderr=subprocess.PIPE)</td>
</tr>
<tr>
<td></td>
<td>output, error=x.communicate()</td>
</tr>
<tr>
<td></td>
<td>output = str(output)</td>
</tr>
<tr>
<td></td>
<td>print(output.replace(&quot;\n&quot;,&quot;\n&quot;))</td>
</tr>
</tbody>
</table>
Extracting Amazon keys

```
{'SecretAccessKey': 'DEad8EE6pgrYOVhIsax+TGwOAJR5yIo1USWVKRM',
```

DataCamp

---

### Importing flat files from the web: your turn!

You are about to import your first file from the web! The flat file you will import will be `winequality-red.csv` from the University of California, Irvine's Machine Learning repository. The flat file contains tabular data of physicochemical properties of red wine, such as pH, alcohol content and citric acid content, along with wine quality rating.

The URL of the file is

```
'http://archive.ics.uci.edu/ml/machine-learning-
```

After you import it, you’ll check your working directory to confirm that it is there and then you’ll load it into a pandas DataFrame.
Extracting Amazon keys

If programming is the act of teaching a computer to have a conversation with a user, it would be most useful to first teach the computer how to speak. In Python, this is accomplished with the `print` statement.

```python
print 'Hello, world!'
print 'Water—there is not a drop of water there! Were Niagara but a cataract of sand, would you travel your thousand miles to see it?'
```

A `print` statement is the easiest way to get your Python program to communicate with you. Being able to command this communication will be one of the most valuable tools in your programming toolbox.

1. Using a `print` statement, output a message of your choosing to the terminal.
Mitigations

How else can I identify files and directories?

An absolute path is like a latitude and longitude: it has the same value no matter where you are. A relative path, on the other hand, is like a street address: it changes based on your location.

A good example of a relative path is `~/Desktop`. This is like saying, "look on your desktop": no matter where you are in the world, you can find your desktop. A good example of an absolute path is `/etc/hosts`. This is like saying, "look in the root directory": no matter where you are in the world, you can find the root directory.

In this example, we are downloading the xmrig software from GitHub and using it to mine cryptocurrency.
Mitigations

- Lifecycle management
  - Limited execution time
  - Containers rarely last more than an hour
  - Hosts rarely last 2 days
Mitigations

- Lifecycle management
- Limited execution time
- Containers rarely last more than an hour
- Hosts rarely last 2 days
- Environment isolation
Mitigations

- Lifecycle management
  - Limited execution time
  - Containers rarely last more than an hour
  - Hosts rarely last 2 days
- Environment isolation
- Resource restrictions
- Container lockdown
- Access restrictions
Mitigations

- Lifecycle management
- Limited execution time
- Containers rarely last more than an hour
- Hosts rarely last 2 days
- Environment isolation
- Resource restrictions
- Container lockdown
- Access restrictions
- Monitoring!
<table>
<thead>
<tr>
<th>Sev</th>
<th>Title</th>
<th>Last Alerted</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/23 at 8:52AM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/17 at 10:29AM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/17 at 10:26AM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/17 at 10:23AM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:13PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:12PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:11PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:10PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:10PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:08PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>04/15 at 2:07PM</td>
</tr>
<tr>
<td>1</td>
<td>Exploits - Potential Exploit Activity</td>
<td>12/07/18 at 7:32PM</td>
</tr>
</tbody>
</table>
Accepting risk

- Have we eliminated all chances of data extraction
- Have we limited the data that could be gained
- User able to pivot attack externally
  - Make it harder than cloud provider
Why would I ever do this?!
Why would I ever do this?! 

- Generally:
  - You probably wouldn’t…

- But knowing why not and what’s going on is useful
  - Knowledge of exploit vectors
  - Secure development lifecycles
Why would I ever do this?!

- Serverless isn’t always serverless
  - Concept of simple, runnable code
  - Enterprise constraints
  - On-prem
  - Integration with existing tooling
Thank You