Unikernels and another way of secure cloud computing

Motiejus Jakštys
motiejus@amazon.com
@mo_kelione

amazon web services

2015-11-19
Disclaimer

This presentation is intended to give a high level overview of the subject matter and is intended for discussion purposes. This presentation is not intended to provide an exhaustive analysis of the subject matter and may differ depending on individual use cases.
# Table of Contents

## Introduction
- Roles of an IT organization
- How to deploy?

## Meat of Unikernels
- Look ma, no OS
- What’s in a unikernel?
- Performance benefits
- Security benefits

## Unikernel examples

## What’s next?
- Unmodified applications first
- Local demo
- In Amazon EC2

## Summary
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td></td>
</tr>
<tr>
<td>Operator Fred Flintstone</td>
<td></td>
</tr>
<tr>
<td>Security certifier Mr Burns</td>
<td></td>
</tr>
</tbody>
</table>
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>▶  Cares only about the app.</td>
</tr>
<tr>
<td>Operator Fred Flintstone</td>
<td></td>
</tr>
<tr>
<td>Security certifier Mr Burns</td>
<td></td>
</tr>
</tbody>
</table>
Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>▶ Cares only about the app.</td>
</tr>
<tr>
<td></td>
<td>▶ Familiar environment (POSIX API).</td>
</tr>
<tr>
<td>Operator Fred Flintstone</td>
<td></td>
</tr>
<tr>
<td>Security certifier Mr Burns</td>
<td></td>
</tr>
</tbody>
</table>
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>- Cares only about the app.</td>
</tr>
<tr>
<td></td>
<td>- Familiar environment (POSIX API).</td>
</tr>
<tr>
<td></td>
<td>- Here’s a JAR, make this thing work.</td>
</tr>
<tr>
<td>Operator Fred Flintstone</td>
<td></td>
</tr>
<tr>
<td>Security certifier Mr Burns</td>
<td></td>
</tr>
</tbody>
</table>
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>▶ Cares only about the app.</td>
</tr>
<tr>
<td></td>
<td>▶ Familiar environment (POSIX API).</td>
</tr>
<tr>
<td></td>
<td>▶ Here’s a JAR, make this thing work.</td>
</tr>
<tr>
<td></td>
<td>▶ Hopefully prod matches dev.</td>
</tr>
<tr>
<td>Operator Fred Flintstone</td>
<td></td>
</tr>
<tr>
<td>Security certifier Mr Burns</td>
<td></td>
</tr>
</tbody>
</table>
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer</strong></td>
<td>▶ Cares only about the app.</td>
</tr>
<tr>
<td></td>
<td>▶ Familiar environment (POSIX API).</td>
</tr>
<tr>
<td></td>
<td>▶ Here’s a JAR, make this thing work.</td>
</tr>
<tr>
<td></td>
<td>▶ Hopefully prod matches dev.</td>
</tr>
<tr>
<td><strong>Operator Fred Flintstone</strong></td>
<td>▶ Simple systems.</td>
</tr>
<tr>
<td><strong>Security certifier Mr Burns</strong></td>
<td></td>
</tr>
</tbody>
</table>
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
</table>
| Developer                        | ▶ Cares only about the app.  
▶ Familiar environment (POSIX API).  
▶ Here’s a JAR, make this thing work.  
▶ Hopefully prod matches dev. |
| Operator Fred Flintstone         | ▶ Simple systems.  
▶ Predictable systems. |
| Security certifier Mr Burns       |                                                                             |
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
</table>
| Developer                     | ▶ Cares only about the app.  
                              | ▶ Familiar environment (POSIX API).  
                              | ▶ Here’s a JAR, make this thing work.  
                              | ▶ Hopefully prod matches dev.        |
| Operator Fred Flintstone      | ▶ Simple systems.  
                              | ▶ Predictable systems.  
                              | ▶ Speedy deployments and rollbacks.  |
| Security certifier Mr Burns   |              |
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
</table>
| Developer                   | ▶ Cares only about the app.  
  ▶ Familiar environment (POSIX API).  
  ▶ Here’s a JAR, make this thing work.  
  ▶ Hopefully prod matches dev.       |
| Operator Fred Flintstone    | ▶ Simple systems.  
  ▶ Predictable systems.  
  ▶ Speedy deployments and rollbacks. |
| Security certifier Mr Burns | ▶ Auditable components.                                                      |
Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Developer</td>
<td>▶ Cares only about the app.</td>
</tr>
<tr>
<td></td>
<td>▶ Familiar environment (POSIX API).</td>
</tr>
<tr>
<td></td>
<td>▶ Here’s a JAR, make this thing work.</td>
</tr>
<tr>
<td></td>
<td>▶ Hopefully prod matches dev.</td>
</tr>
<tr>
<td>Operator Fred Flintstone</td>
<td>▶ Simple systems.</td>
</tr>
<tr>
<td></td>
<td>▶ Predictable systems.</td>
</tr>
<tr>
<td></td>
<td>▶ Speedy deployments and rollbacks.</td>
</tr>
<tr>
<td>Security certifier Mr Burns</td>
<td>▶ Auditable components.</td>
</tr>
<tr>
<td></td>
<td>▶ Least possible dependencies.</td>
</tr>
</tbody>
</table>
## Typical IT organization

<table>
<thead>
<tr>
<th>Role</th>
<th>What system?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Developer</strong></td>
<td>▶ Cares only about the app.</td>
</tr>
<tr>
<td></td>
<td>▶ Familiar environment (POSIX API).</td>
</tr>
<tr>
<td></td>
<td>▶ Here’s a JAR, make this thing work.</td>
</tr>
<tr>
<td></td>
<td>▶ Hopefully prod matches dev.</td>
</tr>
<tr>
<td><strong>Operator Fred Flintstone</strong></td>
<td>▶ Simple systems.</td>
</tr>
<tr>
<td></td>
<td>▶ Predictable systems.</td>
</tr>
<tr>
<td></td>
<td>▶ Speedy deployments and rollbacks.</td>
</tr>
<tr>
<td><strong>Security certifier Mr Burns</strong></td>
<td>▶ Auditable components.</td>
</tr>
<tr>
<td></td>
<td>▶ Least possible dependencies.</td>
</tr>
<tr>
<td></td>
<td>▶ Trades convenience for security.</td>
</tr>
</tbody>
</table>
## How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>mutable, full OS.</td>
<td>▶ Ansible/Chef/Puppet</td>
<td>▶ Adhoc shell scripts</td>
<td>▶ AWS OpsWorks</td>
</tr>
<tr>
<td>unhappy</td>
<td>doesn’t care</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| immutable, full OS. | ▶ Docker | ▶ AMI per release | | |
| happy | doesn’t care | | |
# How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
## How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
# How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td></td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
# How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>- Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>- AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
## How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, full OS.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
# How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, full OS.</td>
<td>Happy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, full OS.</td>
<td>Happy</td>
<td>Unhappy</td>
<td></td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deployment</td>
<td>Developer</td>
<td>Operations</td>
<td>Security</td>
</tr>
<tr>
<td>---------------------</td>
<td>-----------</td>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, full OS.</td>
<td>Happy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, full OS.</td>
<td>Happy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, no OS.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Unikernels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mutable, full OS.</td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, full OS.</td>
<td>Happy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Immutable, no OS.</td>
<td>Happy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Unikernels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mutable, full OS.</strong></td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Immutable, full OS.</strong></td>
<td>Happy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Immutable, no OS.</strong></td>
<td>Happy</td>
<td>Happy</td>
<td></td>
</tr>
<tr>
<td>▶ Unikernels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## How to deploy?

<table>
<thead>
<tr>
<th>Deployment</th>
<th>Developer</th>
<th>Operations</th>
<th>Security</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Mutable, full OS.</strong></td>
<td>Unhappy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Ansible/Chef/Puppet</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ Adhoc shell scripts</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AWS Opsworks</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Immutable, full OS.</strong></td>
<td>Happy</td>
<td>Unhappy</td>
<td>Doesn’t care</td>
</tr>
<tr>
<td>▶ Docker</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>▶ AMI per release</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Immutable, no OS.</strong></td>
<td>Happy</td>
<td>Happy</td>
<td>Happier</td>
</tr>
<tr>
<td>▶ Unikernels</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table of Contents

Introduction
  Roles of an IT organization
  How to deploy?

Meat of Unikernels
  Look ma, no OS
  What’s in a unikernel?
  Performance benefits
  Security benefits

Unikernel examples

What’s next?
  Unmodified applications first
  Local demo
  In Amazon EC2

Summary

Thanks
Unikernel - Library OS

Traditional VM          Unikernel
Unikernel - Library OS

<table>
<thead>
<tr>
<th>Traditional VM</th>
<th>Unikernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration files</td>
<td></td>
</tr>
</tbody>
</table>
## Unikernel - Library OS

<table>
<thead>
<tr>
<th>Traditional VM</th>
<th>Unikernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration files</td>
<td></td>
</tr>
<tr>
<td>Application code</td>
<td></td>
</tr>
</tbody>
</table>
Unikernel - Library OS

Traditional VM

- Configuration files
- Application code
- Language runtime

Unikernel

- Configuration files
- Application code
- Language runtime
Unikernel - Library OS

Traditional VM                  Unikernel

| Configuration files | Application code | Language runtime | User processes |

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
Unikernel - Library OS

<table>
<thead>
<tr>
<th>Traditional VM</th>
<th>Unikernel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Configuration files</td>
<td></td>
</tr>
<tr>
<td>Application code</td>
<td></td>
</tr>
<tr>
<td>Language runtime</td>
<td></td>
</tr>
<tr>
<td>User processes</td>
<td></td>
</tr>
<tr>
<td>Kernel threads</td>
<td></td>
</tr>
<tr>
<td>File System</td>
<td></td>
</tr>
</tbody>
</table>
Unikernel - Library OS

Traditional VM

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack

Unikernel
Unikernel - Library OS

Traditional VM | Unikernel

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack

Hypervisor, x86
Unikernel - Library OS

Traditional VM

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack

Unikernel

- Application code
- Hypervisor, x86

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
Unikernel - Library OS

Traditional VM

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack

Unikernel

- Application code
- Language runtime

Hypervisor, x86
Unikernel - Library OS

Traditional VM

Unikernel

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack
- Hypervisor, x86

- Application code
- Language runtime
- Unikernel runtime
Unikernel - Library OS

Traditional VM

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack

Unikernel

- Application code
- Language runtime
- Unikernel runtime

Hypervisor, x86
Unikernel - Library OS

Traditional VM

- Configuration files
- Application code
- Language runtime
- User processes
- Kernel threads
- File System
- Network stack

Unikernel

- Application code
- Language runtime
- Unikernel runtime

Hypervisor, x86
Introduction
Roles of an IT organization
How to deploy?

**Meat of Unikernels**
Look ma, no OS
What’s in a unikernel?
Performance benefits
Security benefits

Unikernel examples

What’s next?
Unmodified applications first
Local demo
In Amazon EC2

Summary
Thanks
Unikernel has:

- One process, N threads.
Unikernel has:

- One process, N threads.
  - Haha! no context switches.
Unikernel has:

- One process, N threads.
  - Haha! no context switches.
- API for doing network and IO.
Unikernel has:

- One process, N threads.
  - Haha! no context switches.
- API for doing network and IO.
- Possibly application runtime:
Unikernel has:

- One process, N threads.
  - Haha! no context switches.
- API for doing network and IO.
- Possibly application runtime:
Where does performance come from?
Where does performance come from?

Traditional style

NIC memory

Unikernel style

NIC memory

unikernel memory

interrupt

(Developer happy)
Where does performance come from?

Traditional style

NIC memory

\[\text{interrupt}\]

kernel memory

\[\text{(Developer happy)}\]
Where does performance come from?

Traditional style

NIC memory
\[ \text{interrupt} \]
kernel memory
\[ \text{read()} \]
application memory

Unikernel style

NIC memory
unikernel memory
interrupt
is application memory
(Developer happy)
Where does performance come from?

**Traditional style**

1. NIC memory
2. `interrupt`
3. Kernel memory
4. `read()`
5. Application memory

**Unikernel style**

1. NIC memory
2. Unikernel memory
3. `interrupt`
4. Is application memory (Developer happy)
Where does performance come from?

**Traditional style**

- NIC memory
  - interrupt
  - kernel memory
  - read()
  - application memory

**Unikernel style**

- NIC memory

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
Where does performance come from?

**Traditional style**

NIC memory  \[\downarrow\] interrupt

kernel memory  \[\downarrow\] read()

application memory

**Unikernel style**

NIC memory  \[\downarrow\] interrupt

unikernel memory
Where does performance come from?

Traditional style

NIC memory

\[\text{interrupt}\]

kernel memory

\[\text{read()}\]

application memory

Unikernel style

NIC memory

\[\text{interrupt}\]

unikernel memory

is application memory
Where does performance come from?

Traditional style

NIC memory
\[\text{interrupt}\]
kernel memory
\[\text{read()}\]
application memory

Unikernel style

NIC memory
\[\text{interrupt}\]
unikernel memory
is application memory
(Developer happy)
Introduction
Roles of an IT organization
How to deploy?

Meat of Unikernels
Look ma, no OS
What’s in a unikernel?
Performance benefits

Security benefits

Unikernel examples

What’s next?
Unmodified applications first
Local demo
In Amazon EC2

Summary

Thanks
Security

- Reduced attack surface area.
Security

- Reduced attack surface area.
  - No unneeded kernel modules.
Security

- Reduced attack surface area.
  - No unneeded kernel modules.
  - No Perl, shell...
Security

- Reduced attack surface area.
  - No unneeded kernel modules.
  - No Perl, shell...
  - Exploited environment is barely usable to useless.
Security

- Reduced attack surface area.
  - No unneeded kernel modules.
  - No Perl, shell...
  - Exploited environment is barely usable to useless.
- Some: language is type-safe.
Security

- Reduced attack surface area.
  - No unneeded kernel modules.
  - No Perl, shell...
  - Exploited environment is barely usable to useless.
- Some: language is type-safe.
- Some: can do 1 VM per request.
Security

- Reduced attack surface area.
  - No unneeded kernel modules.
  - No Perl, shell...
  - Exploited environment is barely usable to useless.
- Some: language is type-safe.
- Some: can do 1 VM per request.

Happy Mr Burns
Why only in 2013?

Virtualization provides uniform APIs for network and I/O.
Why only in 2013?

Virtualization provides uniform APIs for network and I/O.
E.g. *virtio* for KVM, *netfront/netback* for Xen.
- Small set of drivers to implement.
- Makes it economic to create unikernels.
Table of Contents

Introduction
  Roles of an IT organization
  How to deploy?

Meat of Unikernels
  Look ma, no OS
  What’s in a unikernel?
  Performance benefits
  Security benefits

Unikernel examples

What’s next?
  Unmodified applications first
  Local demo
  In Amazon EC2

Summary

Thanks

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
Classification

**Niche**: highly optimized applications/frameworks.
Classification

Niche : highly optimized applications/frameworks.

Generic : general-purpose applications:
Classification

Niche : highly optimized applications/frameworks.

Generic : general-purpose applications:

Fat : yes POSIX. Compatible.
Classification

**Niche** : highly optimized applications/frameworks.

**Generic** : general-purpose applications:
- **Fat** : yes POSIX. Compatible.
- **Lean** : no POSIX. Needs rewrite.
Mirage OS

Generic lean.
Mirage OS

*Generic lean.*

- OCaml-only.

Typical image sizes: 100s of KB.

Typical application: web services, data processing.
Mirage OS

Generic lean.

- OCaml-only.
- \(\Rightarrow\) type-safe throughout the stack.
Mirage OS

Generic lean.

- OCaml-only.
- ⇒ type-safe throughout the stack.
- VM boot time < 10ms
Mirage OS

Generic lean.

- OCaml-only.
- ⇒ type-safe throughout the stack.
- VM boot time $< 10ms$
- ⇒ VM per request (inetd-like).
Mirage OS

Generic lean.

► OCaml-only.
► ⇒ type-safe throughout the stack.
► VM boot time < 10ms
► ⇒ VM per request (inetd-like).
► ⇒ security and economy.
Mirage OS

Generic lean.

- OCaml-only.
- \( \Rightarrow \) type-safe throughout the stack.
- VM boot time \(<\ 10\text{ms}\)
- \( \Rightarrow \) VM per request (inetd-like).
- \( \Rightarrow \) security and economy.
- Typical image sizes: 100s of KB.
Mirage OS

Generic lean.

- OCaml-only.
- ⇒ type-safe throughout the stack.
- VM boot time < 10ms
- ⇒ VM per request (inetd-like).
- ⇒ security and economy.
- Typical image sizes: 100s of KB.

Typical application: web services, data processing.
Generic Fat.
Generic Fat.

- Linux ABIs.

Bonus: fast APIs.

Memory allocation tuning:
- JVM.
- Memcached.

OOM Killer Manager!

Subsecond boot times.

Typical application: web services, data processing.
Generic Fat.

- Linux ABIs.
- Bonus: fast APIs.
**Generic Fat.**

- Linux ABIs.
- **Bonus:** fast APIs.
  - reported significant throughput increases on memcached.
Generic Fat.

- Linux ABIs.
- Bonus: fast APIs.
  - reported significant throughput increases on memcached.
- Memory allocation tuning:
Generic Fat.

- Linux ABIs.
- Bonus: fast APIs.
  - reported significant throughput increases on memcached.
- Memory allocation tuning:
  - JVM.
**Generic Fat.**

- Linux ABIs.
- **Bonus:** fast APIs.
  - reported significant throughput increases on memcached.
- Memory allocation tuning:
  - JVM.
  - Memcached.
Generic Fat.

- Linux ABIs.
- Bonus: fast APIs.
  - reported significant throughput increases on memcached.
- Memory allocation tuning:
  - JVM.
  - Memcached.
  - OOM Killer Manager!
Generic Fat.

- Linux ABIs.
- Bonus: fast APIs.
  - reported significant throughput increases on memcached.
- Memory allocation tuning:
  - JVM.
  - Memcached.
  - OOM Killer Manager!
- Subsecond boot times.
OSv

Generic Fat.

- Linux ABIs.
- Bonus: fast APIs.
  - reported significant throughput increases on memcached.
- Memory allocation tuning:
  - JVM.
  - Memcached.
  - OOM Killer Manager!
- Subsecond boot times.

Typical application: web services, data processing.
ClickOS

Niche.
Optimized for network middleboxes:
▶ Firewalls.
▶ Intrusion Detection Systems.
▶ Load Balancers.

Fun fact: Xen network optimization to the extreme.
ClickOS

Niche.
ClickOS

Niche.
Optimized for network middleboxes:
Niche.
Optimized for network middleboxes:
  ► Firewalls.
ClickOS

*Niche.*

Optimized for network middleboxes:

- Firewalls.
- Intrusion Detection Systems.
Niche.
Optimized for network middleboxes:
  - Firewalls.
  - Intrusion Detection Systems.
  - Load Balancers.
ClickOS

*Niche.*
Optimized for network middleboxes:
- Firewalls.
- Intrusion Detection Systems.
- Load Balancers.

Fun fact: Xen network optimization to the extreme.
What do we do next?

Nobody likes rewrites.
What do we do next?

Nobody likes rewrites.
Take platforms (JVM, Erlang) and run unmodified apps.
  ▶ Fully immutable.
What do we do next?

Nobody likes rewrites.
Take platforms (JVM, Erlang) and run unmodified apps.

- Fully immutable.
- Faster deployment and rollbacks.
What do we do next?

Nobody likes rewrites.
Take platforms (JVM, Erlang) and run unmodified apps.

- Fully immutable.
- Faster deployment and rollbacks.
- Smaller attack surface.
Local demo

Contents:

▶ Take a JVM/Spring application.
Local demo

Contents:

- Take a JVM/Spring application.
- Demo on standard Linux.
Local demo

Contents:

- Take a JVM/Spring application.
- Demo on standard Linux.
- Generate the unikernel.
Local demo

Contents:
- Take a JVM/Spring application.
- Demo on standard Linux.
- Generate the unikernel.
- Run that unikernel locally.
Local demo

Contents:
- Take a JVM/Spring application.
- Demo on standard Linux.
- Generate the unikernel.
- Run that unikernel locally.

Takeaways:
Local demo

Contents:

- Take a JVM/Spring application.
- Demo on standard Linux.
- Generate the unikernel.
- Run that unikernel locally.

Takeaways:

- Small image size.
Local demo

Contents:

- Take a JVM/Spring application.
- Demo on standard Linux.
- Generate the unikernel.
- Run that unikernel locally.

Takeaways:

- Small image size.
- Trivial to implement.
Running in Amazon EC2

To create a VM image in AWS, do:

```
% qemu-img convert -f qcow2 -O raw 3.qemu 3.raw

% ./release-ec2.sh \
   --override-image 3.raw \
   --override-version 3 \
   --region us-east-1
```
AMI in EC2
AMI in EC2

```
Instance: i-6b8f20d7 (osv3) Public DNS: ec2-52-28-202-224.eu-central-1.compute.amazonaws.com

<table>
<thead>
<tr>
<th>Description</th>
<th>Status Checks</th>
<th>Monitoring</th>
<th>Tags</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance ID</td>
<td>i-6b8f20d7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance state</td>
<td>running</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instance type</td>
<td>c4.large</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private DNS</td>
<td>ip-172.31.4.84.eu-central-1.compute.amazonaws</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Private IPs</td>
<td>172.31.4.84</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VPC ID</td>
<td>vpc-2a23c043</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Subnet ID</td>
<td>subnet-e8a3d890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public DNS</td>
<td>ec2-52-28-202-224.eu-central-1.compute.amazonaws</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Public IP</td>
<td>52.28.202.224</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Elastic IP</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Availability zone</td>
<td>eu-central-1b</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Security groups</td>
<td>allow-http-ssh, view rules</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Scheduled events</td>
<td>No scheduled events</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AMI ID</td>
<td>OSv-3 (ami-05061469)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Platform</td>
<td>windows</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Running OSv in EC2

Greetings from Spring Boot!
Try it out

- Run your unikernel in Free Tier right now.
Try it out

- Run your unikernel in Free Tier right now.
- t2.micro - $0/month for 1 year.
# Table of Contents

**Introduction**
- Roles of an IT organization
- How to deploy?

**Meat of Unikernels**
- Look ma, no OS
- What’s in a unikernel?
- Performance benefits
- Security benefits

**Unikernel examples**

**What’s next?**
- Unmodified applications first
- Local demo
- In Amazon EC2

**Summary**

© 2015. Amazon Web Services, Inc. or its affiliates. All rights reserved.
Library OS: OS embedded to your application.
Summary

Library OS: OS embedded to your application.
- Small $\Rightarrow$ scale quickly.
Summary

Library OS: OS embedded to your application.

- Small $\Rightarrow$ scale quickly.
- Very efficient $\Rightarrow$ economic.
Summary

Library OS: OS embedded to your application.

- Small ⇒ scale quickly.
- Very efficient ⇒ economic.
- Reduced attack surface.
Summary

Library OS: OS embedded to your application.

- Small ⇒ scale quickly.
- Very efficient ⇒ economic.
- Reduced attack surface.
- Runs on public clouds: ⇒ try on EC2, for free.
Thanks

- Niels Brouwers (Amazon) for the right tools.
- Russel Pavlicek (Citrix) for spreading the word.
We’re hiring!

- Check out amazon.jobs
- Also, contact me at motiejus@amazon.com