NYMOTE: GIT YOUR OWN CLOUD HERE

Anil Madhavapeddy University of Cambridge @avsm

Richard Mortier University of Nottingham @mort___

http://openmirage.org/
http://nymote.org/
http://decks.openmirage.org/oscon14/

Press <esc> to view the slide index, and the <arrow> keys to navigate.

These slides were written using Mirage on OSX:

- They are hosted in a **938kB Xen unikernel** written in statically type-safe OCaml, including device drivers and network stack.
- Their application logic is just a **couple of source files**, written independently of any OS dependencies.
- Running on an **ARM** CubieBoard2, and hosted on the cloud.
- Binaries small enough to track the **entire deployment** in Git!

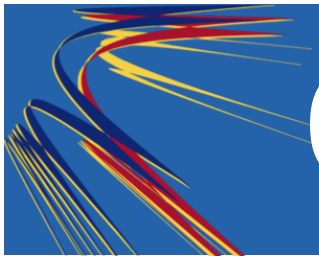
INTRODUCING MIRAGE OS 2.0

LEANING TOWER OF CLOUD

Numerous pain points:

- **Complex** configuration management.
- Duplicated functionality leads to **inefficiency**.
- VM image size leads to **long boot times**.
- Lots of code means a **large attack surface**.

https://flic.kr/p/8N1hWh



COMPLEXITY KILLS YOU

The enemy is **complexity**:

- Applications are **deeply intertwined** with system APIs, and so lack portability.
- Modern operating systems offer **dynamic support** for **many users** to run **multiple applications** simultaneously.

Almost unbounded scope for uncontrolled interaction!

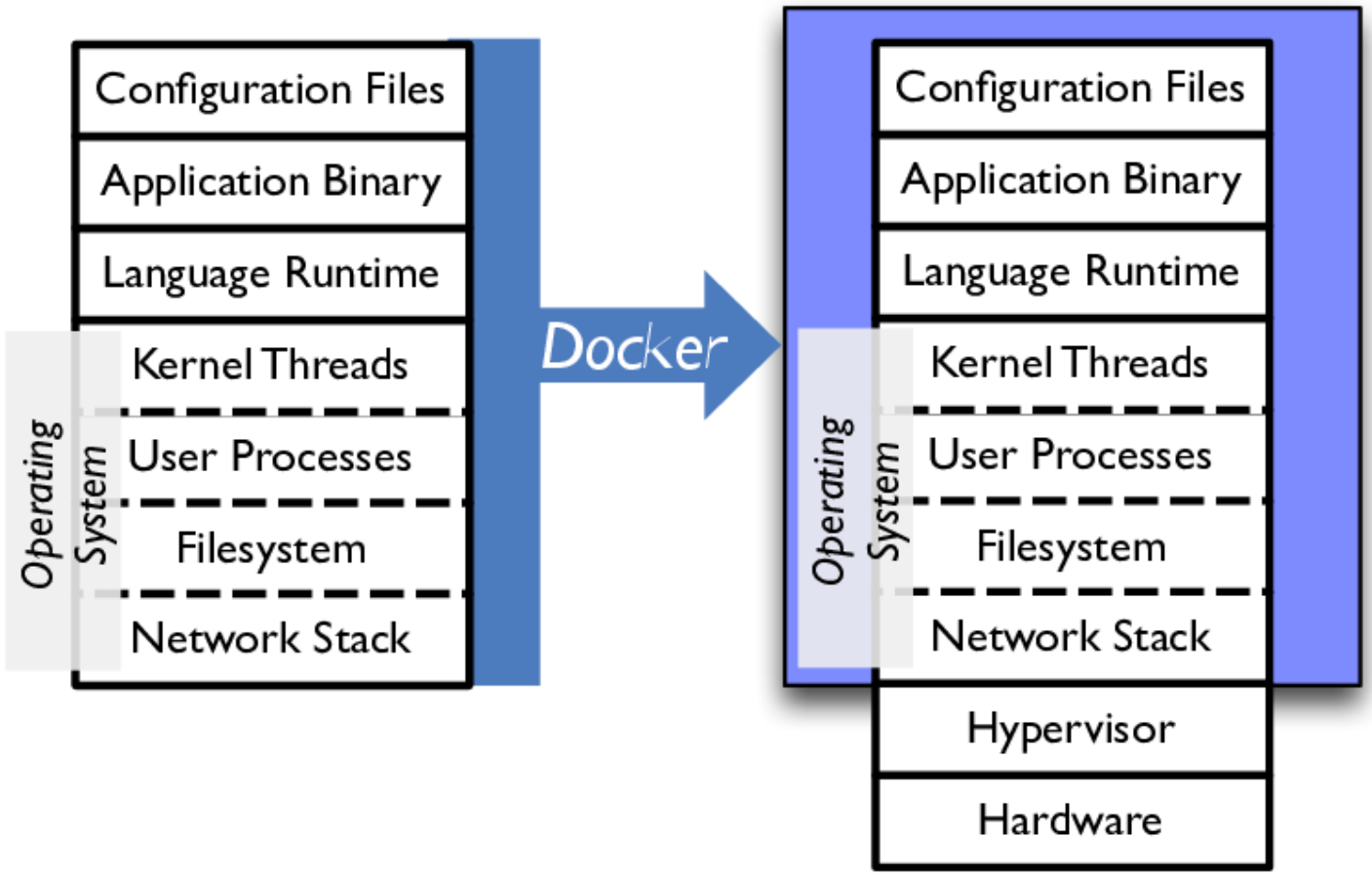
- Choices of distribution and version.
- Ad hoc application configuration under `/etc/`
- Platform configuration details, e.g., firewalls.

DOCKER: CONTAINERISATION

https://flic.kr/p/qSbck

DOCKER: CONTAINERISATION

Docker bundles up all this state making it easy to transport, install and manage.



Disentangle applications from the operating system.

- Break up operating system functionality into modular libraries.
- Link only the system functionality your app needs.
- Target alternative platforms from a single codebase.

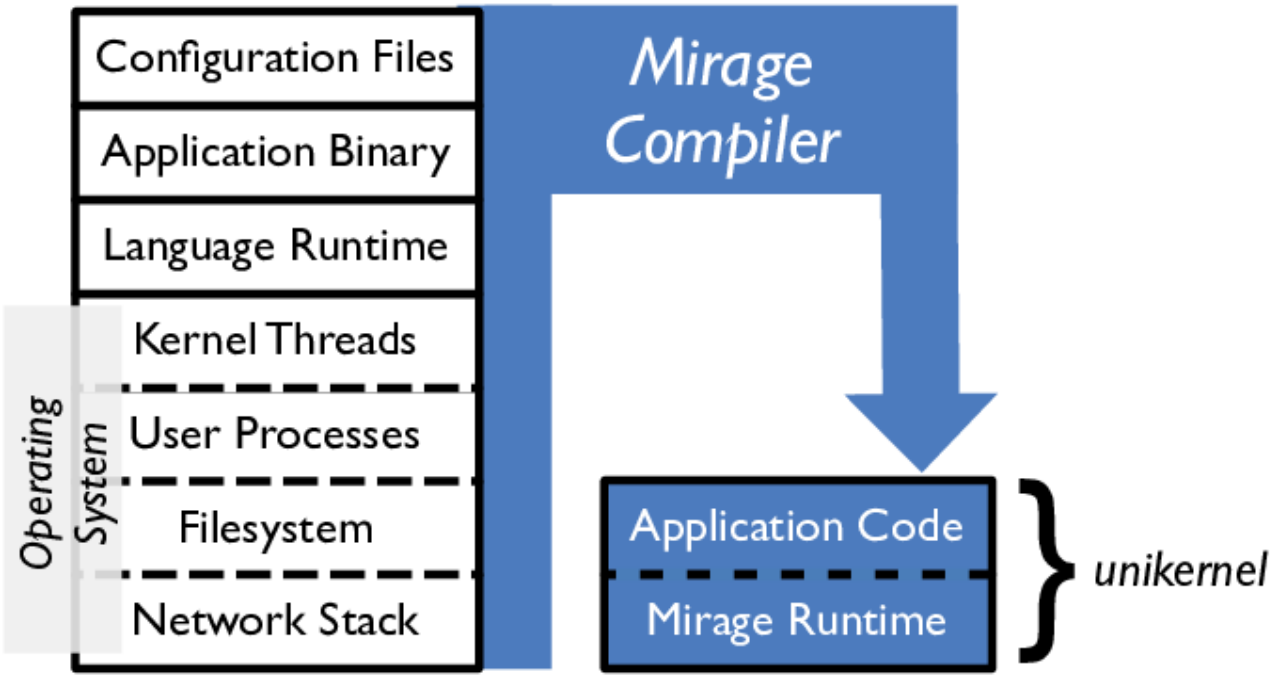
Unikernels are specialised virtual machine images compiled from the full stack of application code, system libraries and config

This means they realise several benefits:

- **Contained**, simplifying deployment and management.
- **Compact**, reducing attack surface and boot times.
- **Efficient**, able to fit 10,000s onto a single host.

IT'S ALL JUST SOURCE CODE

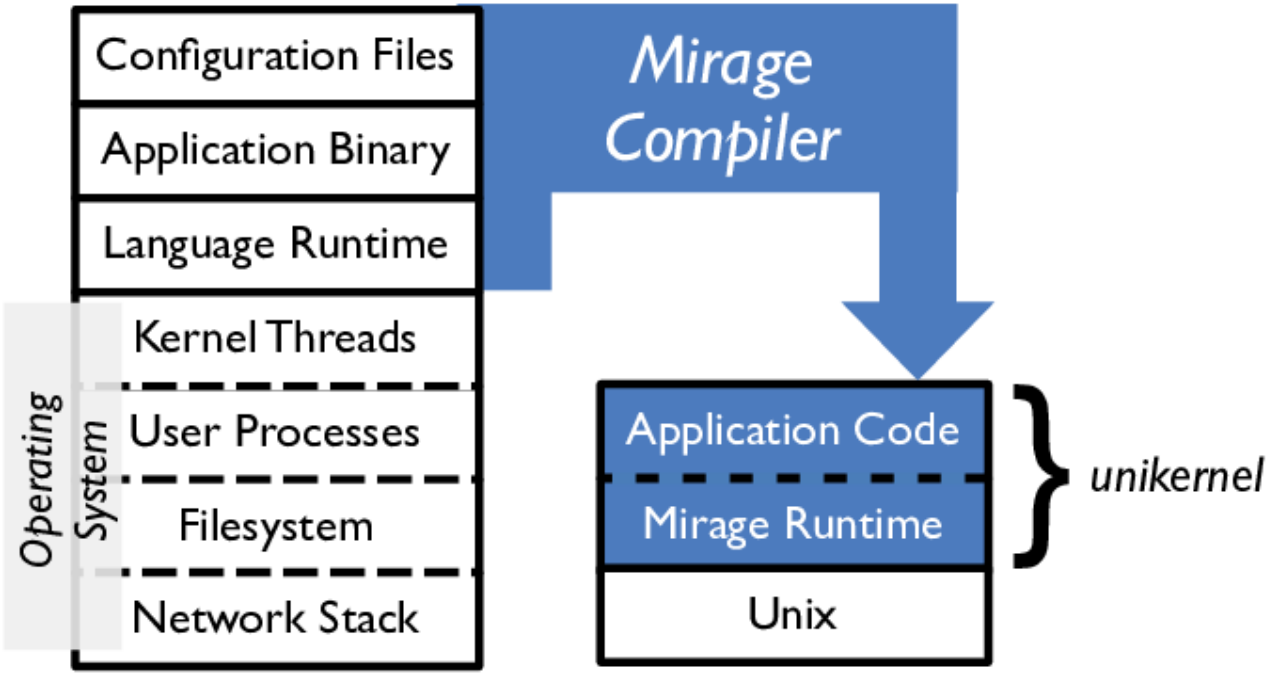
Capture system dependencies in code and compile them away.



RETARGET BY RECOMPILING

Swap system libraries to target different platforms:

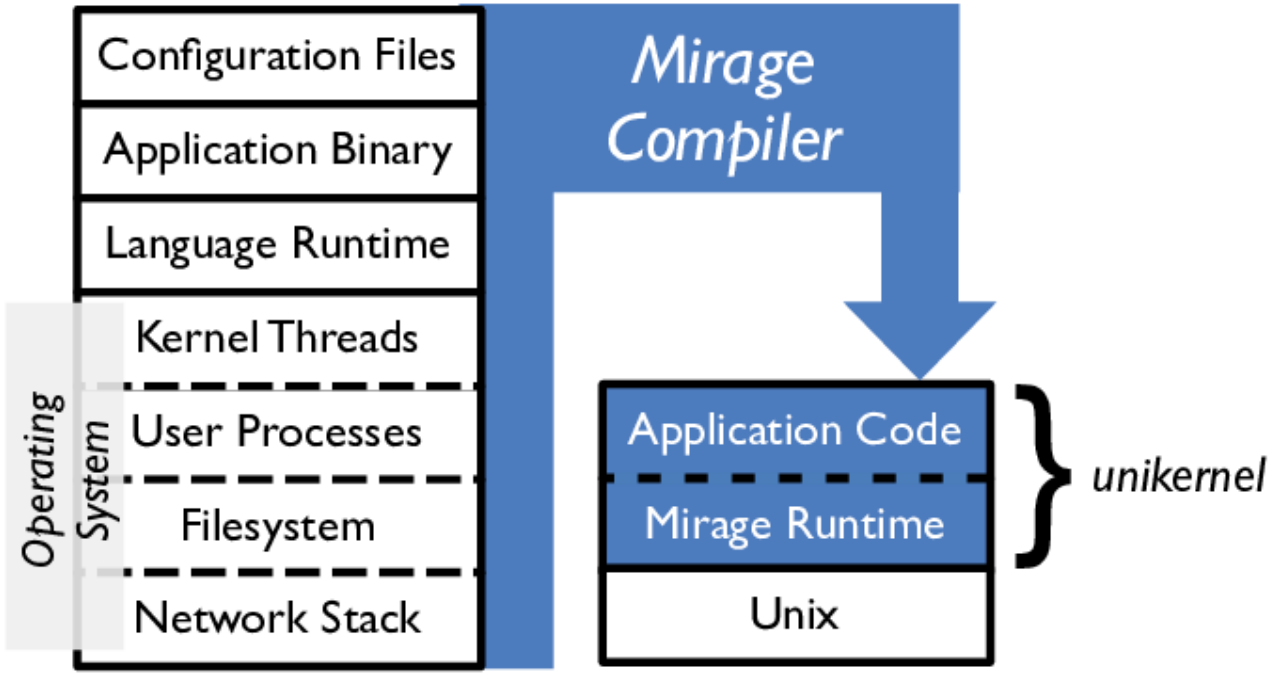
develop application logic using native Unix.



RETARGET BY RECOMPILING

Swap system libraries to target different platforms:

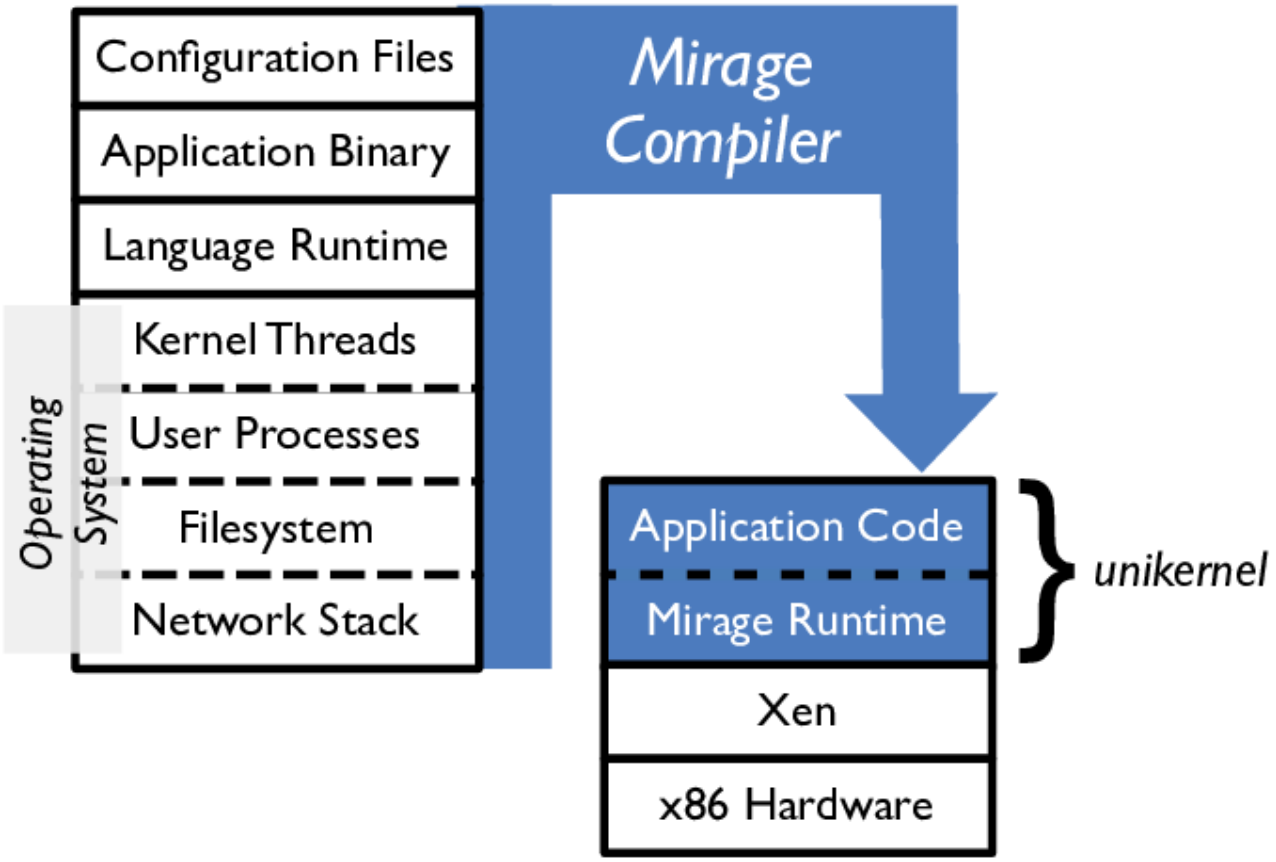
test unikernel using Mirage system libraries.



RETARGET BY RECOMPILING

Swap system libraries to target different platforms:

deploy by specialising unikernel to Xen.

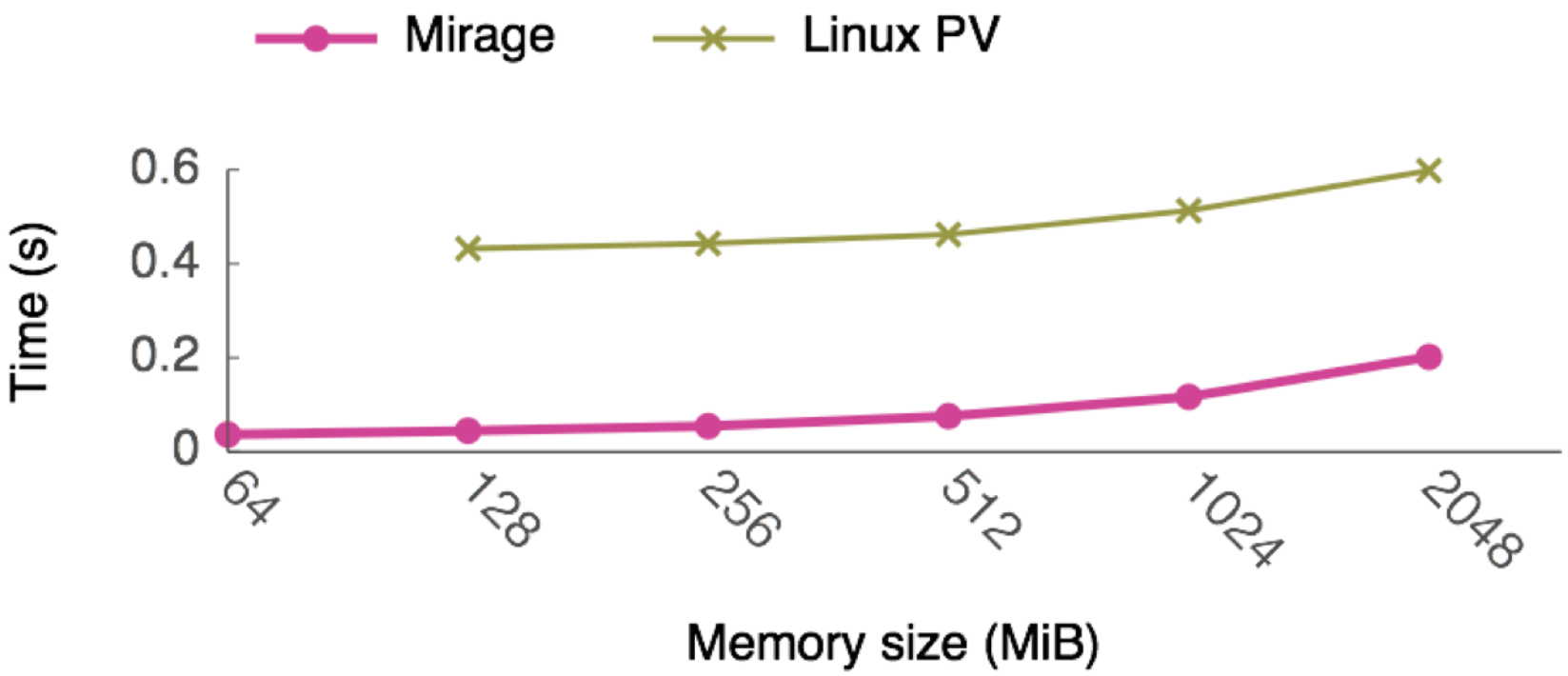


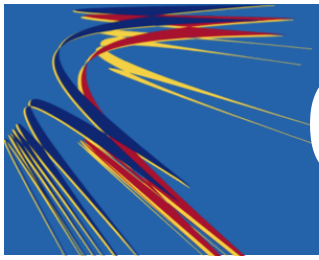
Unikernels are compact enough to boot and respond to network traffic in real-time.

Appliance	Standard Build	Dead Code Elimination
DNS	0.449 MB	0.184 MB
Web Server	0.674 MB	0.172 MB
Openflow learning switch	0.393 MB	0.164 MB
Openflow controller	0.392 MB	0.168 MB

END RESULT?

Unikernels are compact enough to boot and respond to network traffic in real-time.





GIT YOUR OWN CLOUD

Unikernels are **small enough to be tracked in GitHub**. For example, for the Mirage website:

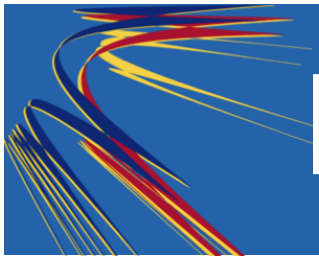
1. Source code updates are merged to mirage/mirage-www;
2. Repository is continuously rebuilt by Travis CI; if successful:
3. Unikernel pushed to mirage/mirage-www-deployment; and
4. Our cloud toolstack spawns VMs based on pushes there.

Our *entire* cloud-facing deployment is version-controlled from the source code up!

As easy as 1—2—3!

1. Write your OCaml application using the Mirage module types.
    - Express its configuration as OCaml code too!

```
$ mirage configure app/config.ml --unix
```

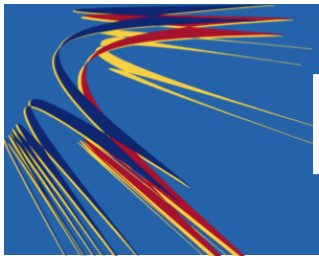


MIRAGE OS 2.0 WORKFLOW

As easy as 1—2—3!

1. Write your OCaml application using the Mirage module types.
   - Express its configuration as OCaml code too!
2. Compile it and debug under Unix using the `mirage` tool.

```
$ cd app
$ make depend # install library dependencies
$ make build  # build the unikernel
```



MIRAGE OS 2.0 WORKFLOW

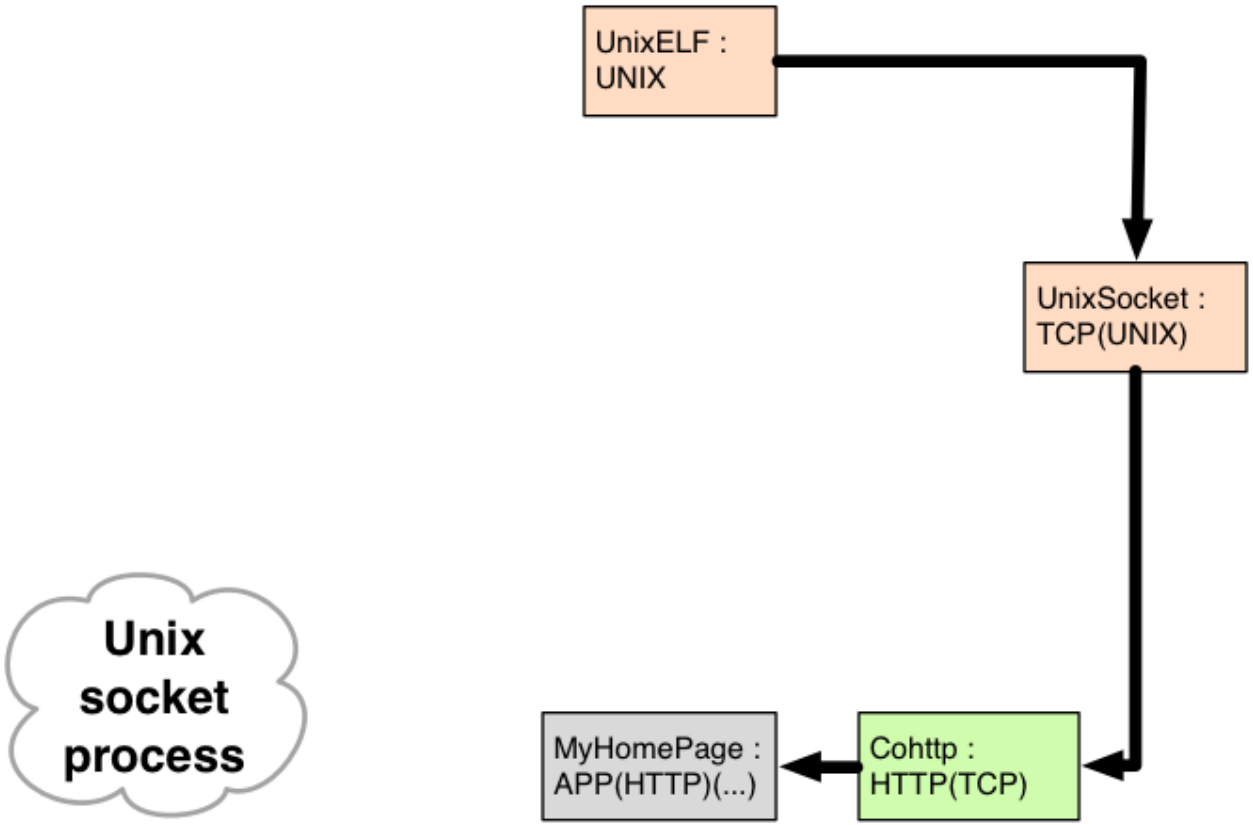
As easy as 1—2—3!

1. Write your OCaml application using the Mirage module types.
   - Express its configuration as OCaml code too!
2. Compile it and debug under Unix using the `mirage` tool.
3. Once debugged, simply retarget it to Xen, and rebuild!

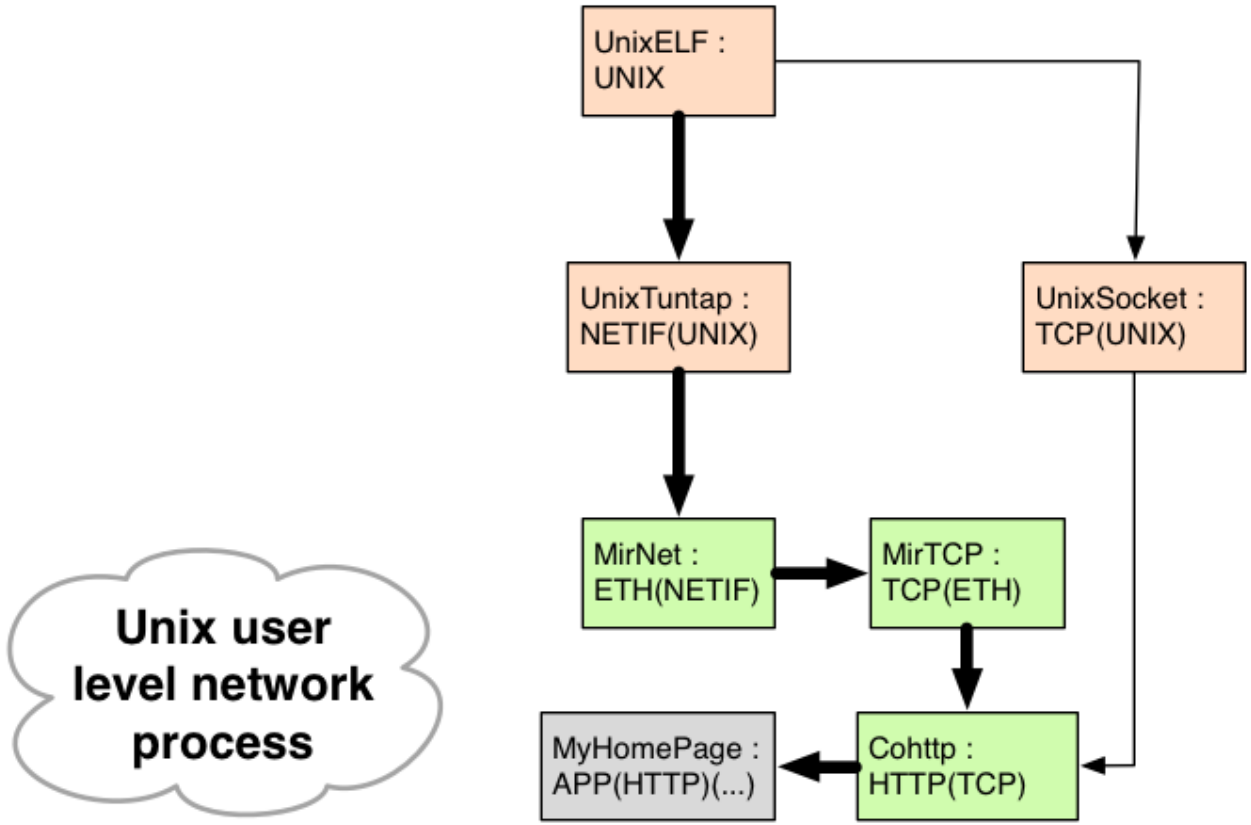
```
$ mirage configure app/config.ml --xen
$ cd app && make depend && make build
```

   - All the magic happens via the OCaml module system.

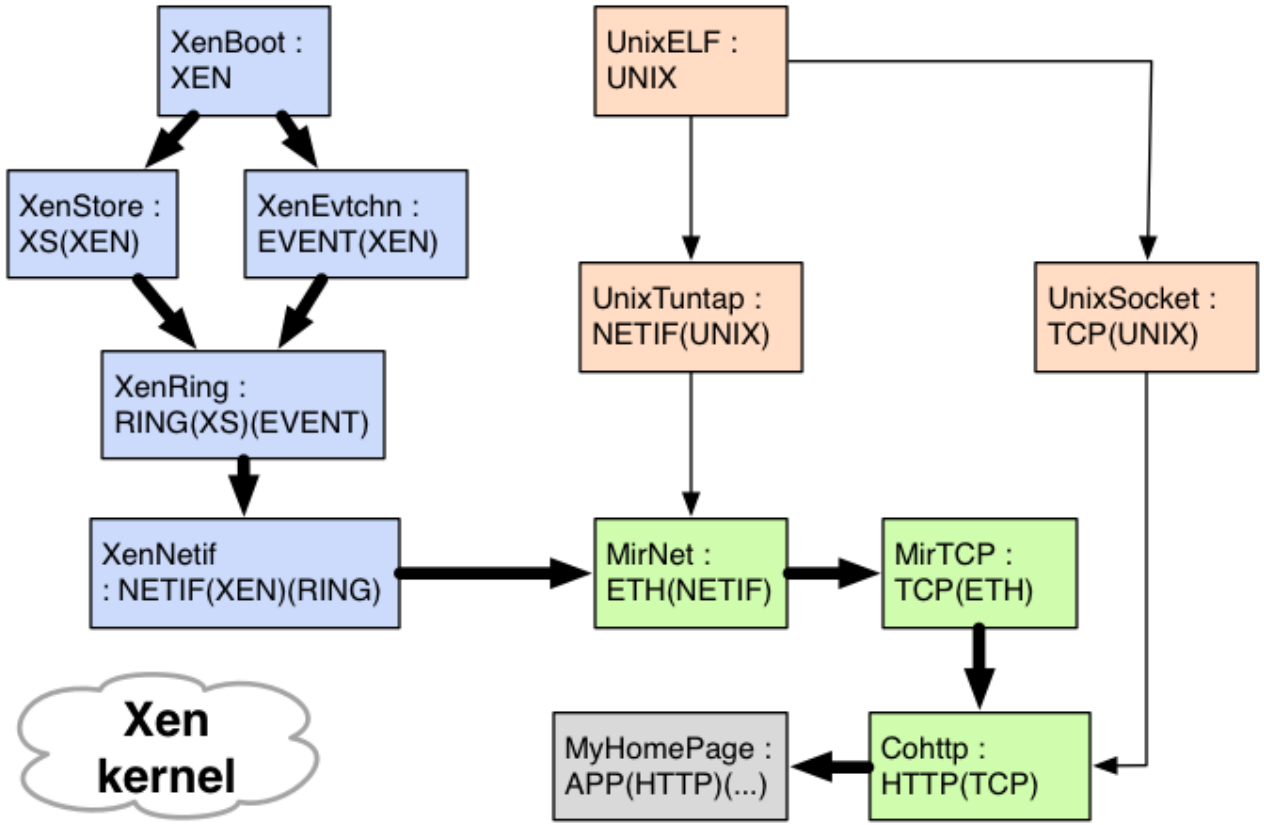
MODULARIZING THE OS

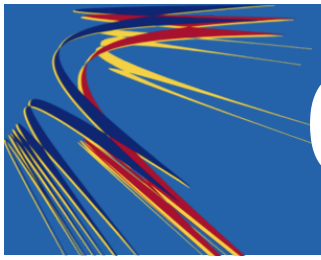


MODULARIZING THE OS



MODULARIZING THE OS



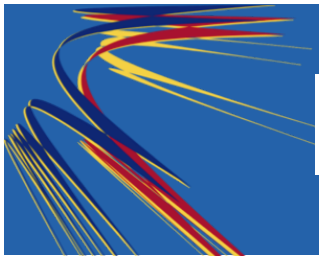


GIT YOUR OWN CLOUD

Unikernels are **small enough to be tracked in GitHub**. For example, for the Mirage website:

1. Source code updates are merged to mirage/mirage-www;
2. Repository is continuously rebuilt by Travis CI; if successful:
3. Unikernel pushed to mirage/mirage-www-deployment; and our
4. Cloud toolstack spawns VMs based on pushes there.

Our *entire* cloud-facing deployment is version-controlled from the source code up!



IMPLICATIONS

Historical tracking of source code and built binaries in Git(hub).

- `git tag` to link code and binary across repositories.
- `git log` to view deployment changelog.
- `git pull` to deploy new version.
- `git checkout` to go back in time to any point.
- `git bisect` to pin down deployment failures.

Historical tracking of source code and built binaries in Git(hub).

Low latency deployment of security updates.

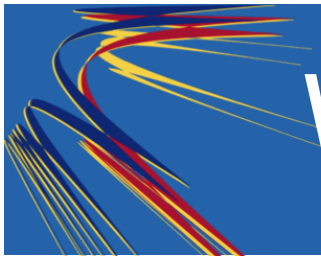
- No need for Linux distro to pick up and build the new version.
- Updated binary automatically built and pushed.
- Pick up latest binary directly from repository.
- Statically type-checked language prevents classes of attack.

Historical tracking of source code and built binaries in Git(hub).

Low latency deployment of security updates.

Unified development for cloud and embedded environments.

- Write application code once.
- Recompile to swap in different versions of system libraries.
- Use compiler optimisations for exotic environments.



WRAPPING UP

Mirage OS 2.0 is an important step forward, supporting **more**, and **more diverse**, **backends** with much **greater modularity**.

For information about the many components we could not cover here, see openmirage.org:

- Irmin, Git-like distributed branchable storage.
- OCaml-TLS, a from-scratch native OCaml TLS stack.
- Vchan, for low-latency inter-VM communication.
- Ctypes, modular C foreign function bindings.

We need to claim control over our online lives rather than abrogate it to *The Cloud*:

- Doing so means we **all** need to be able to run **our own infrastructure**.
- **Without** having to become (Linux) **sysadmins**!

How can we achieve this?

Mirage is the foundation for building **personal clouds**, securely interconnecting and synchronising data between our devices.

Featuring blog posts by: Amir Chaudhry, Thomas Gazagnaire, David Kaloper, Thomas Leonard, Jon Ludlam, Hannes Mehnert, Mindy Preston, Dave Scott, and Jeremy Yallop.

Thanks for listening! Questions?

(*and please rate the talk*!)