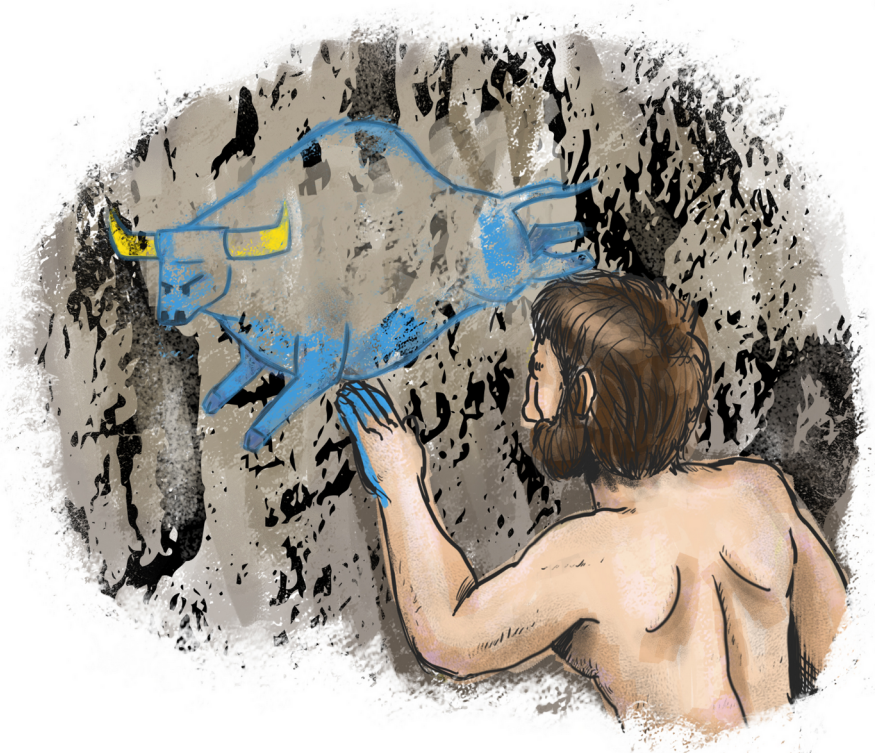
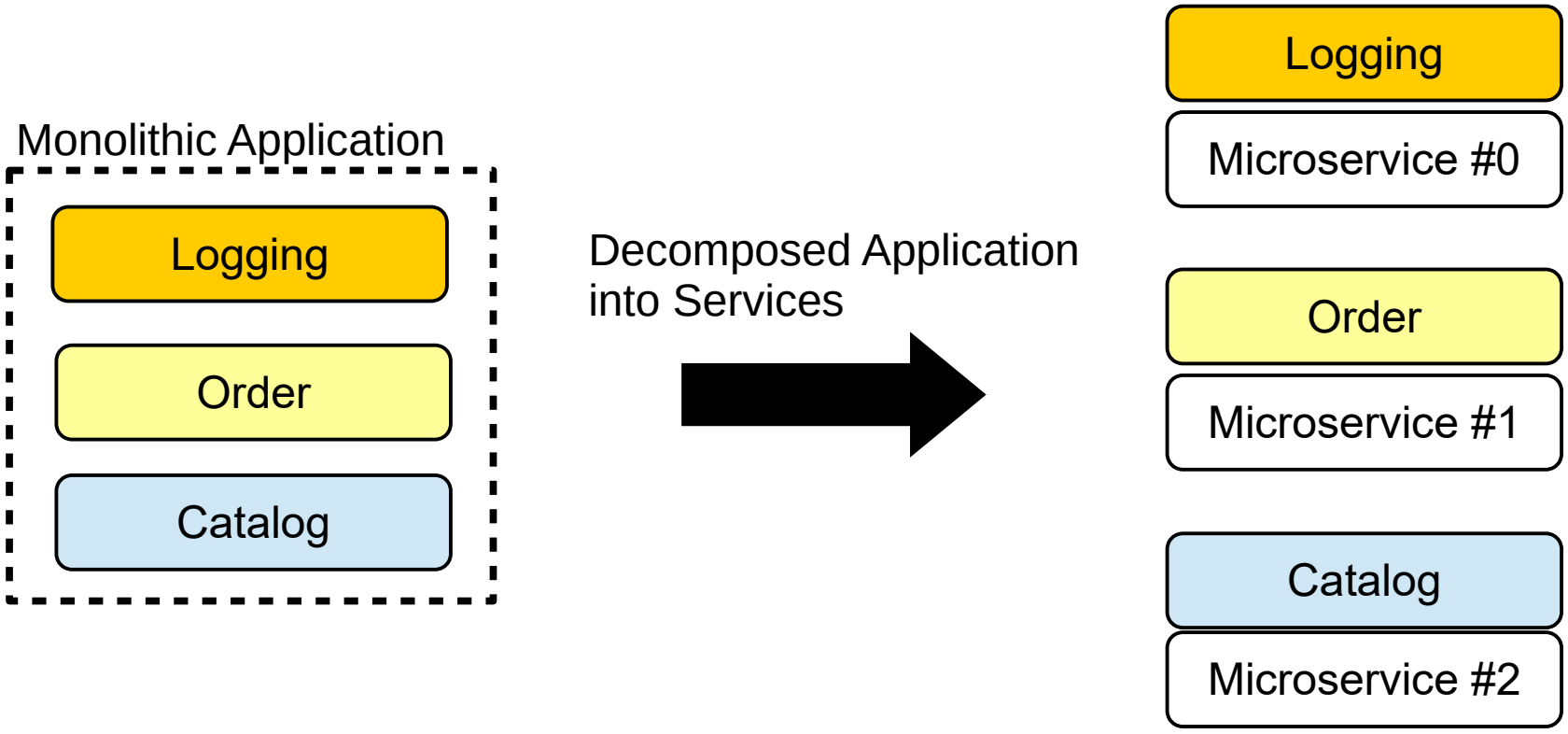


Leveraging Virtio-fs and Virtio-vsocket in Toro Unikernel

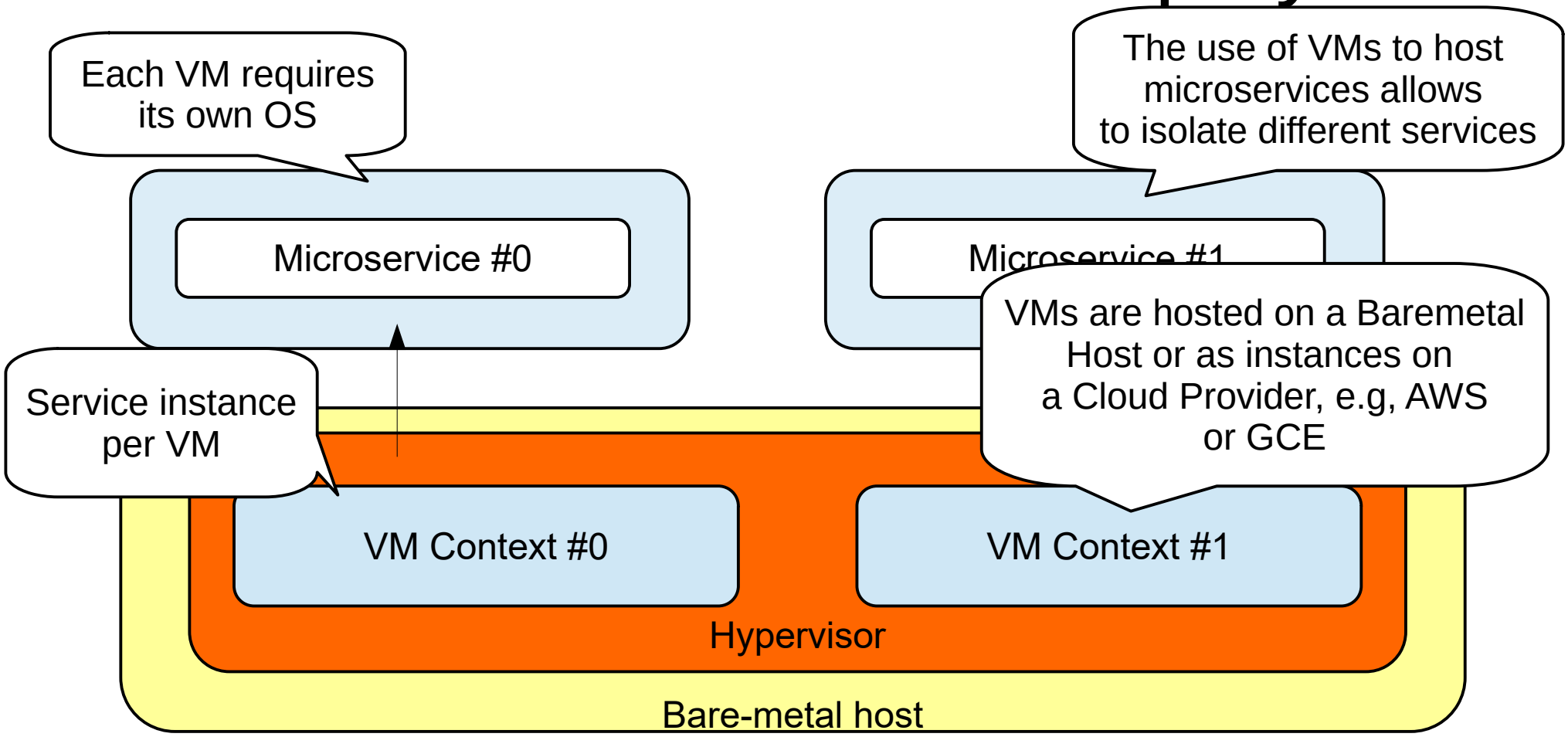
www.torokernel.io

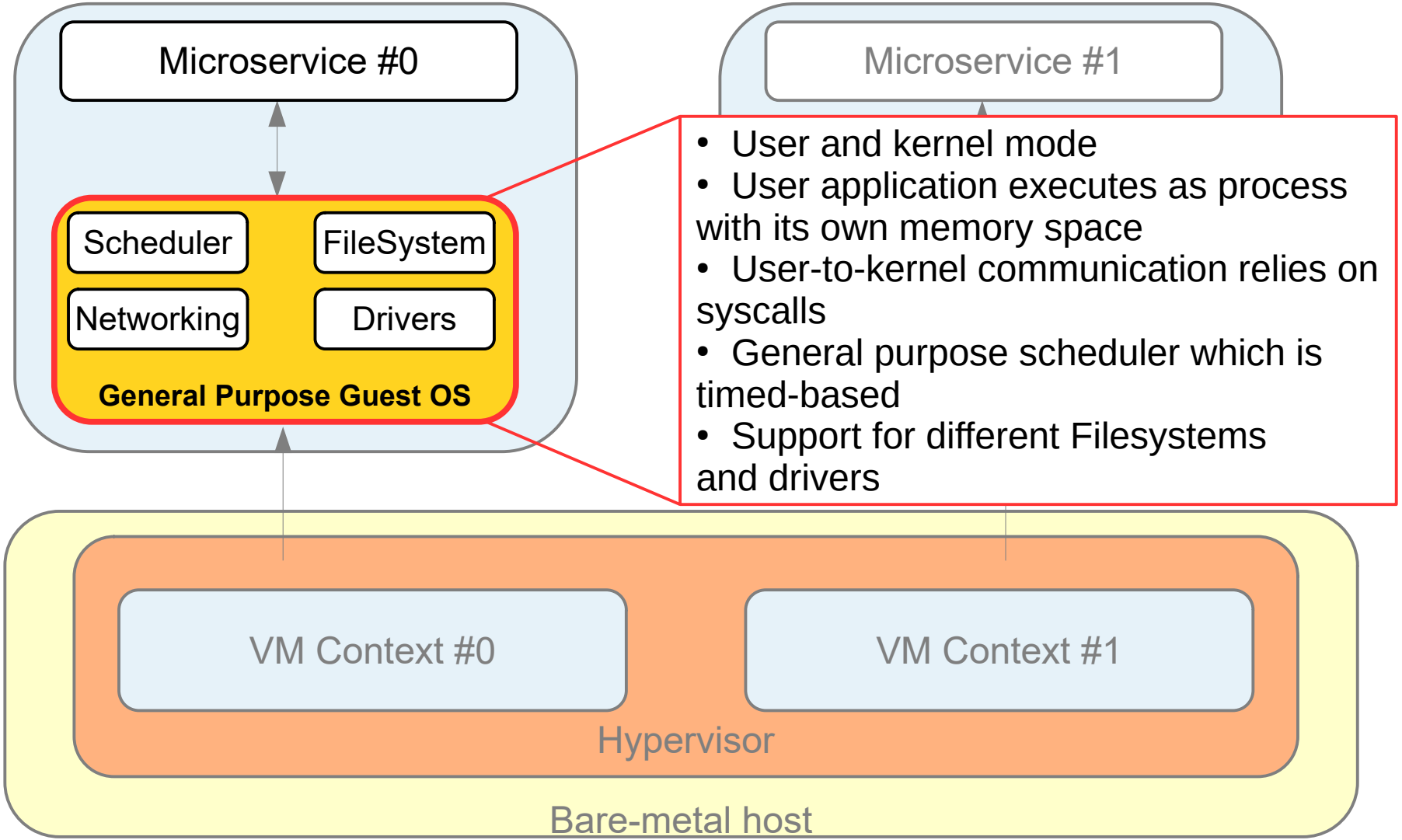
Matias Vara Larsen
matiasevara@gmail.com

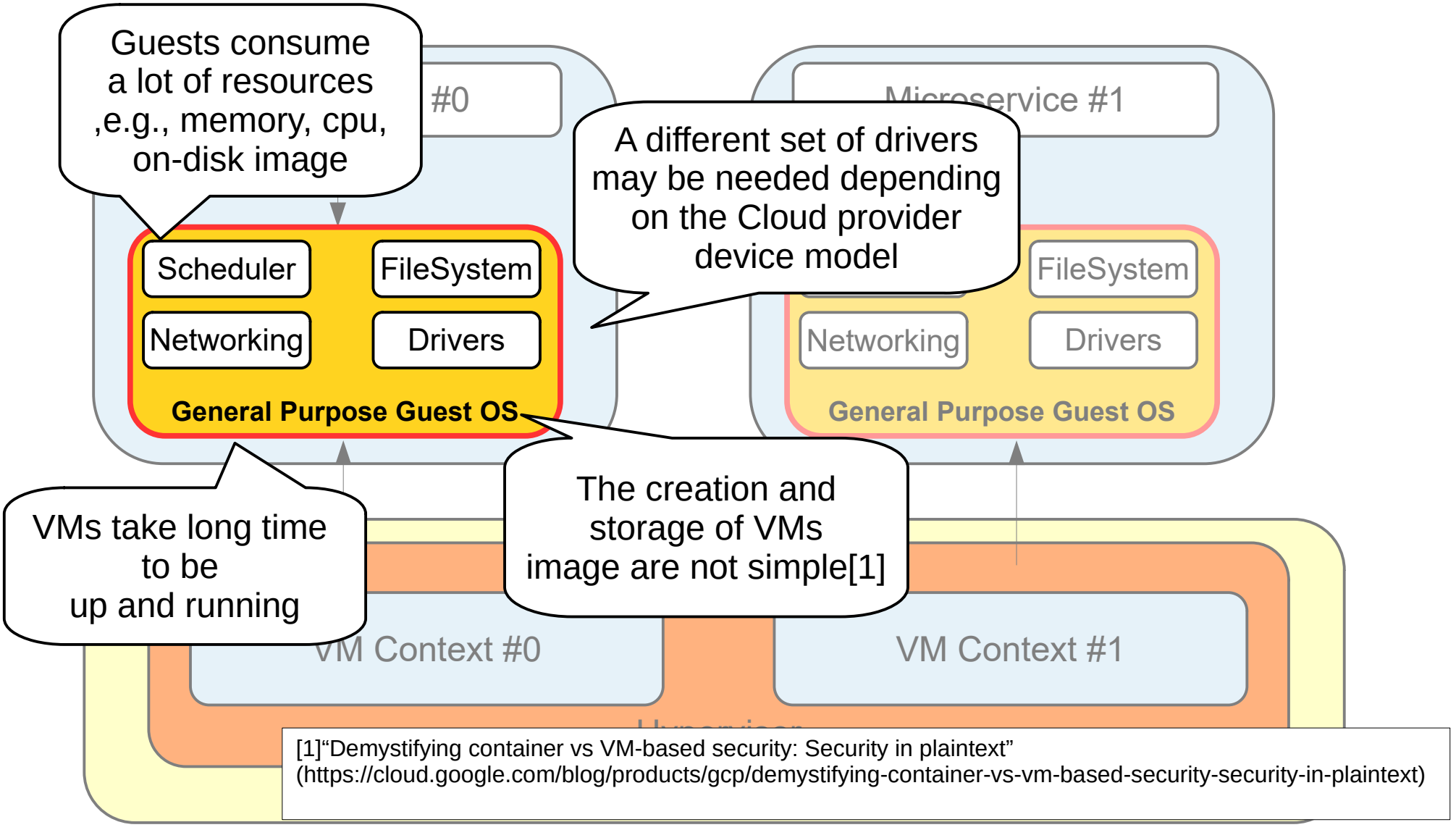
What are microservices?



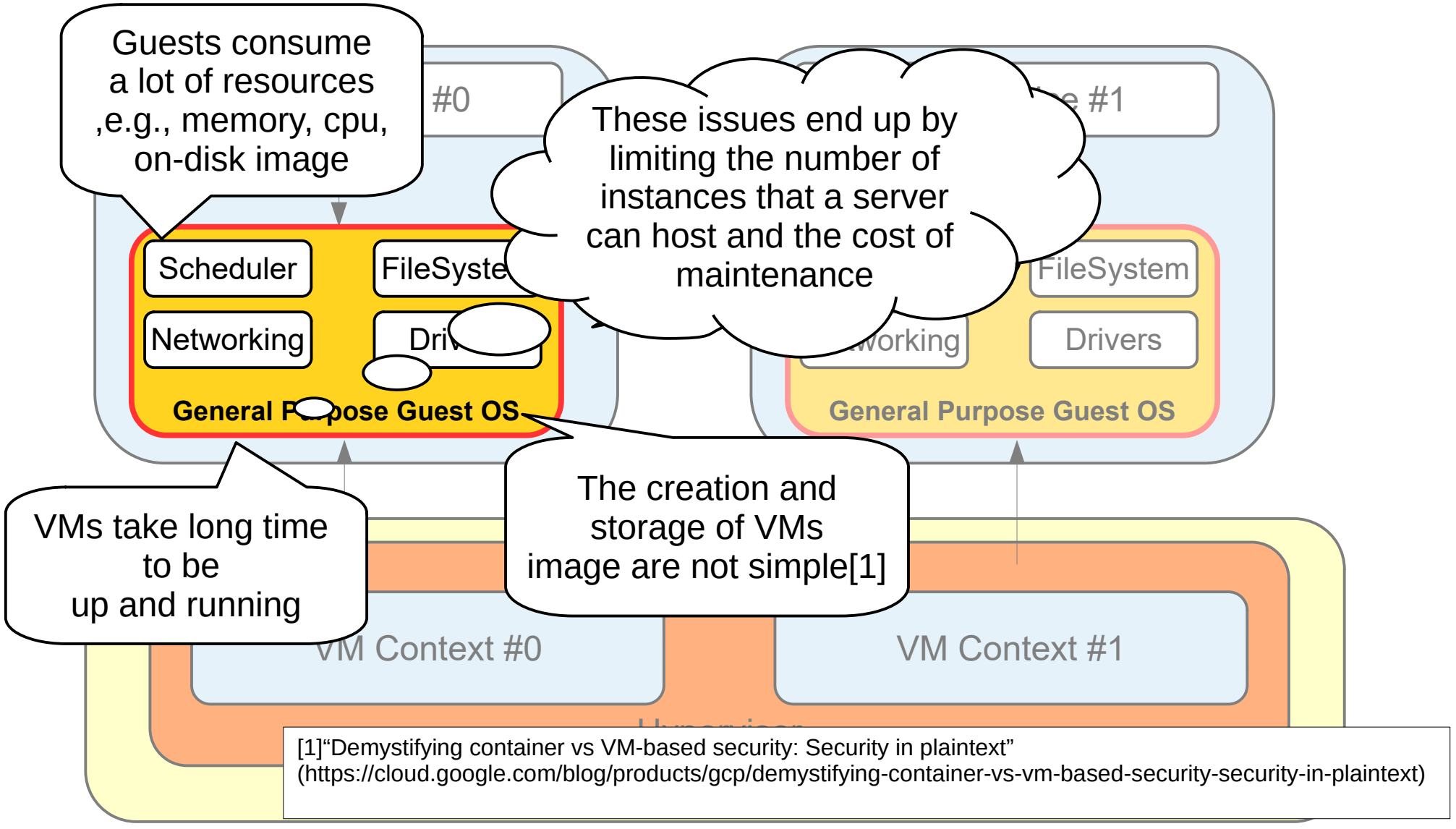
How are microservices deployed?

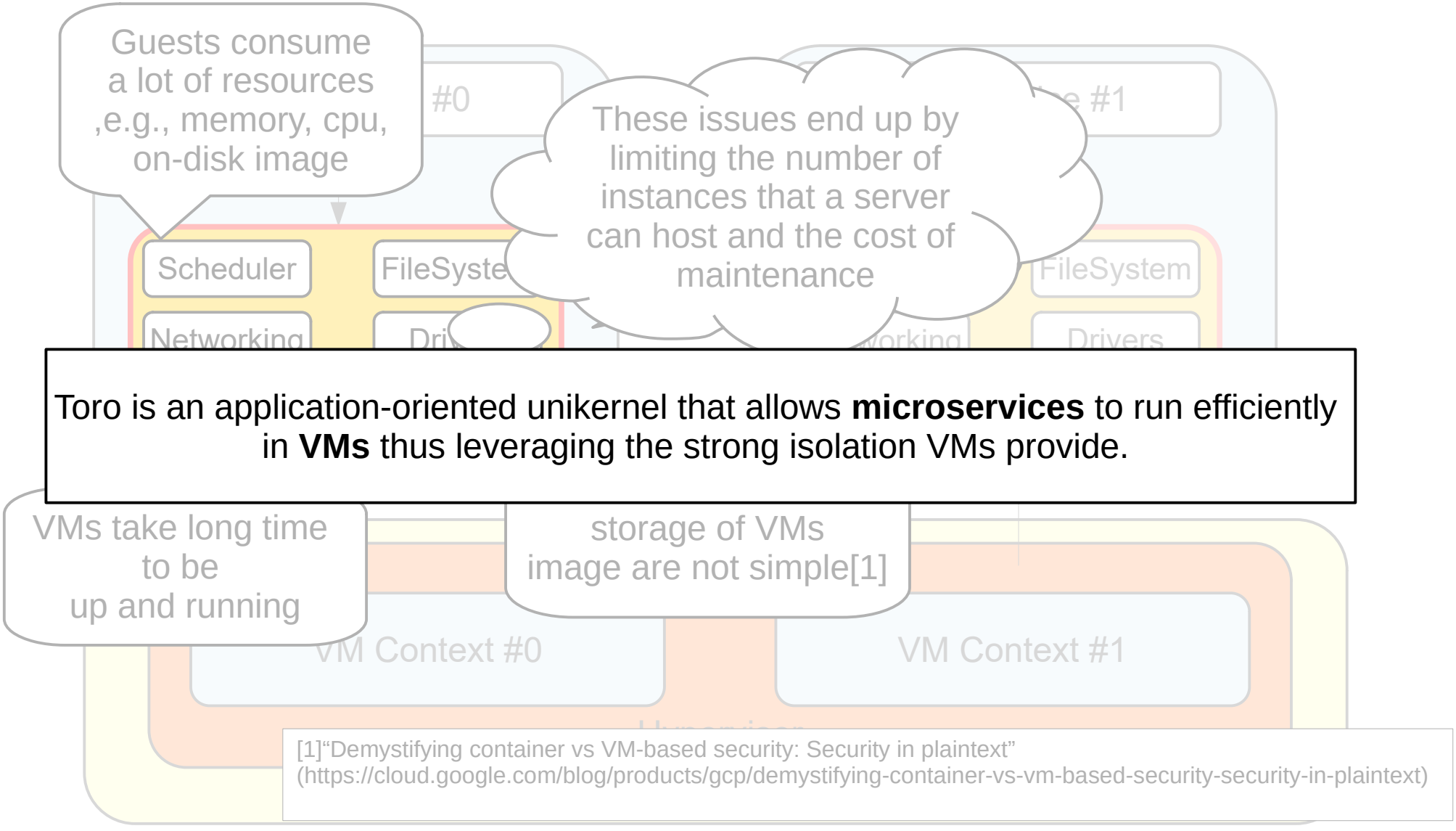



Microservice #0
Microservice #1
Scheduler
FileSystem
Networking
Drivers
General Purpose Guest OS
• User and kernel mode
• User application executes as process with its own memory space
• User-to-kernel communication relies on syscalls
• General purpose scheduler which is timed-based
• Support for different Filesystems and drivers
VM Context #0
VM Context #1
Hypervisor
Bare-metal host



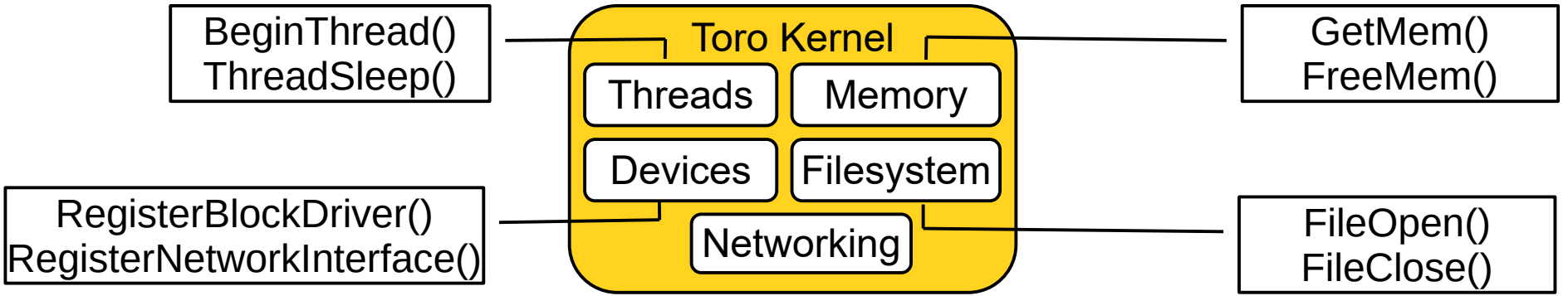
[1]“Demystifying container vs VM-based security: Security in plaintext” (https://cloud.google.com/blog/products/gcp/demystifying-container-vs-vm-based-security-security-in-plaintext)


Guests consume a lot of resources ,e.g., memory, cpu, on-disk image
These issues end up by limiting the number of instances that a server can host and the cost of maintenance
Scheduler
Networking
General Purpose Guest OS
FileSystem
Drivers
General Purpose Guest OS
VMs take long time to be up and running
The creation and storage of VMs image are not simple[1]
VM Context #0
VM Context #1
[1]“Demystifying container vs VM-based security: Security in plaintext” (https://cloud.google.com/blog/products/gcp/demystifying-container-vs-vm-based-security-security-in-plaintext)



[1]“Demystifying container vs VM-based security: Security in plaintext” (https://cloud.google.com/blog/products/gcp/demystifying-container-vs-vm-based-security-security-in-plaintext)

Application-oriented Kernel

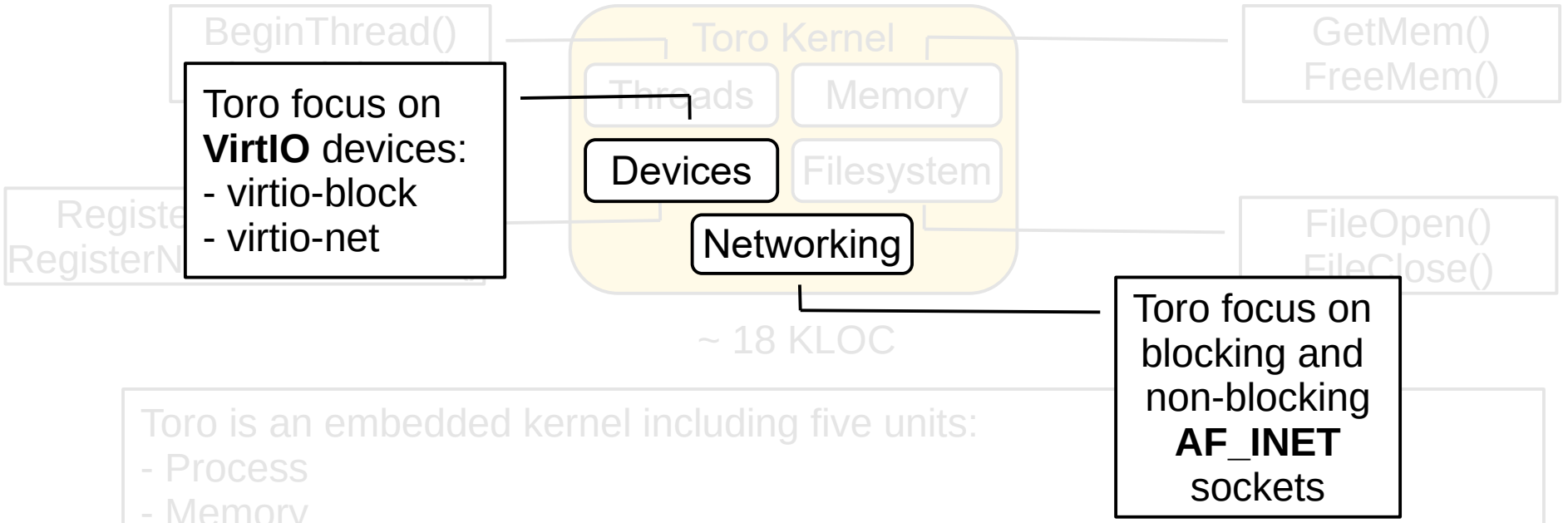


~ 18 KLOC

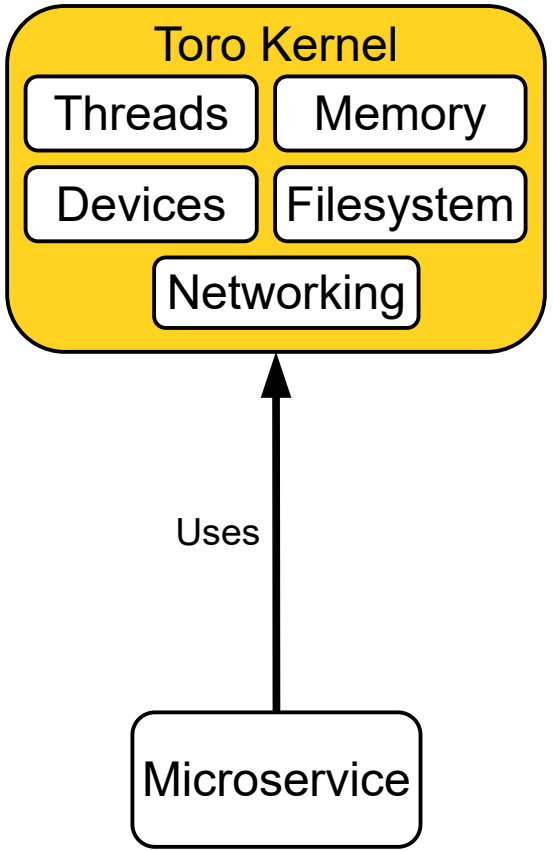
Toro is an embedded kernel including five units:
- Process
- Memory
- Filesystem
- Networking
- Devices, e.g., Block Device, Network Device

Each unit provides minimalist APIs accessible from the embedded application

Application-oriented Kernel

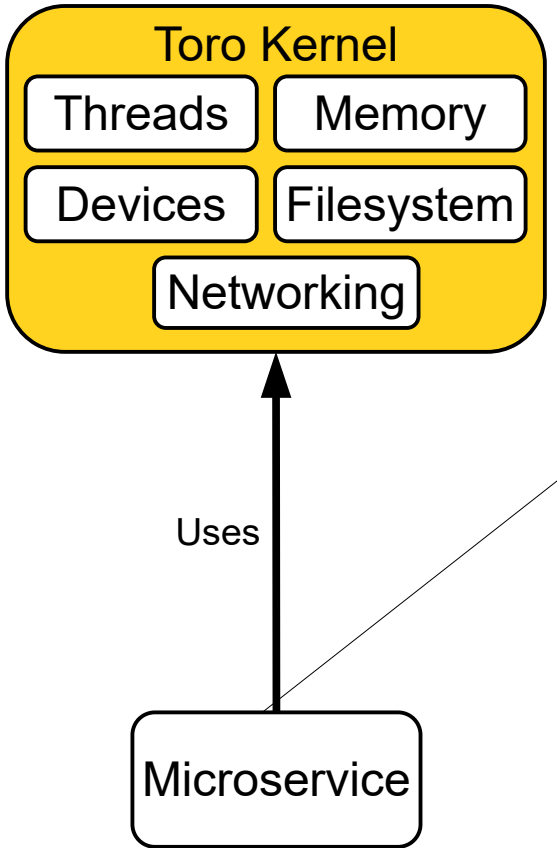


Application-oriented Kernel



- User application and kernel units are compiled in a single binary
- The application includes only the component required

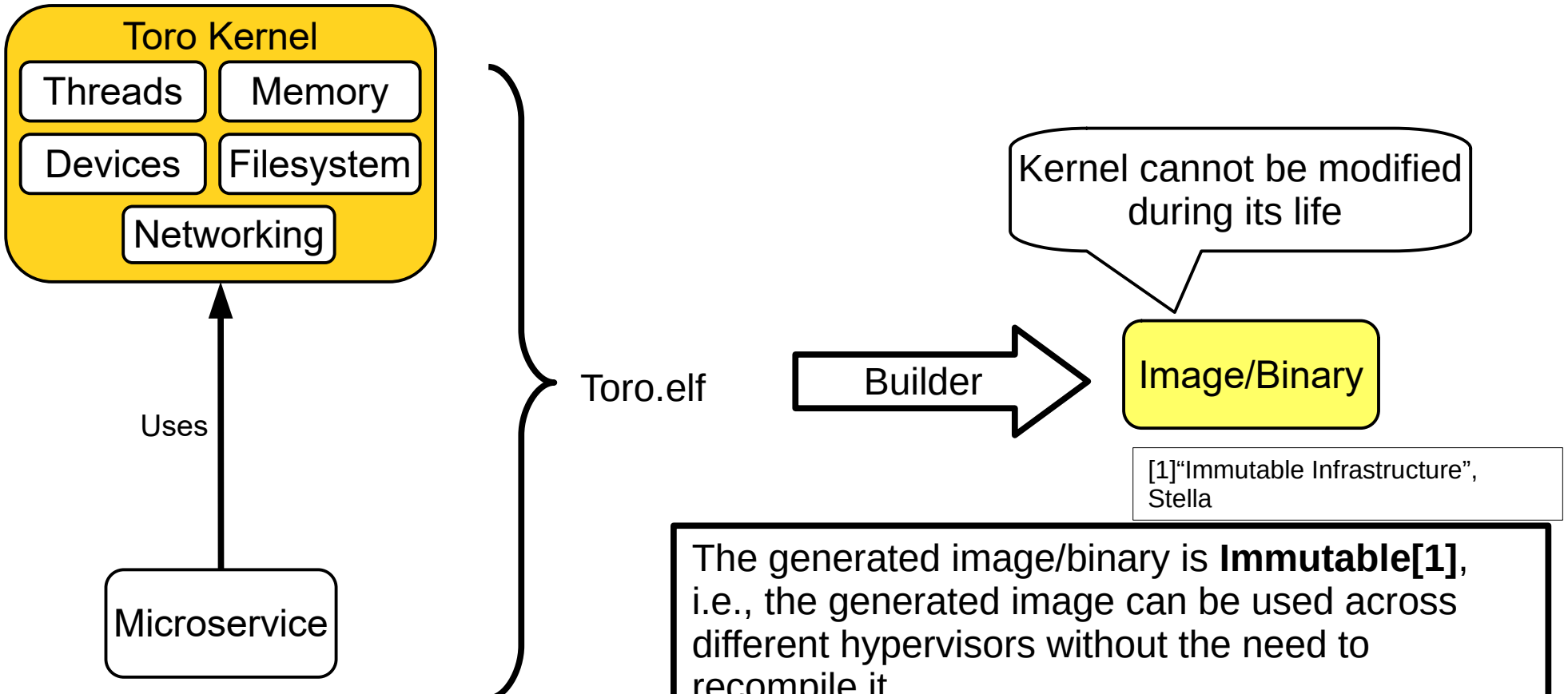
Application-oriented Kernel



- User application and kernel units are compiled in a single binary
- The
com

```
program WebServerAppliance;
uses
  Memory,
  Filesystem,
  Threads,
  Networking,
  Fat,
  Virtio-blk,
  Virtio-net;
Begin
//
// Your Code Goes Here
//
End.
```

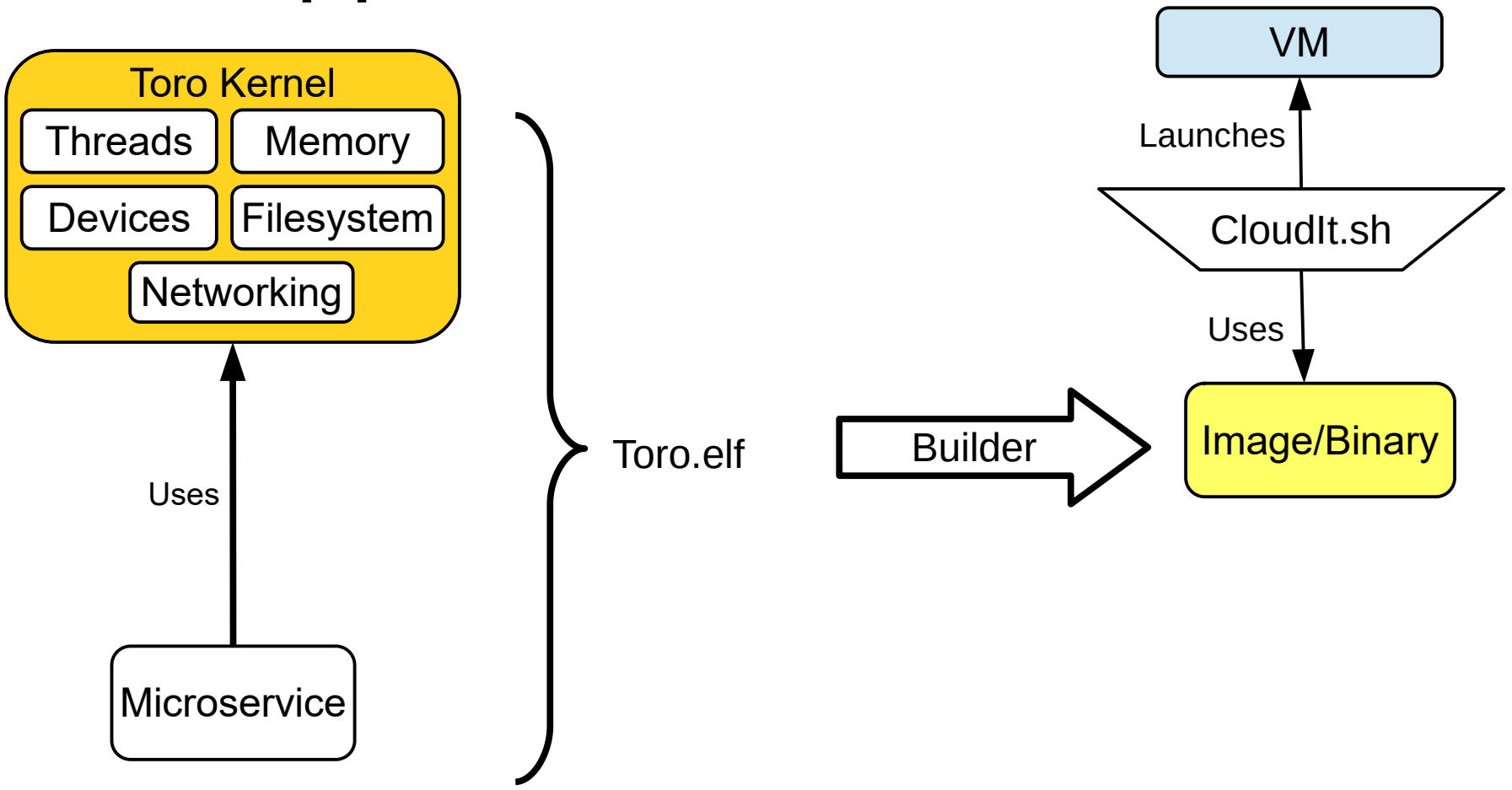
Application-oriented Kernel



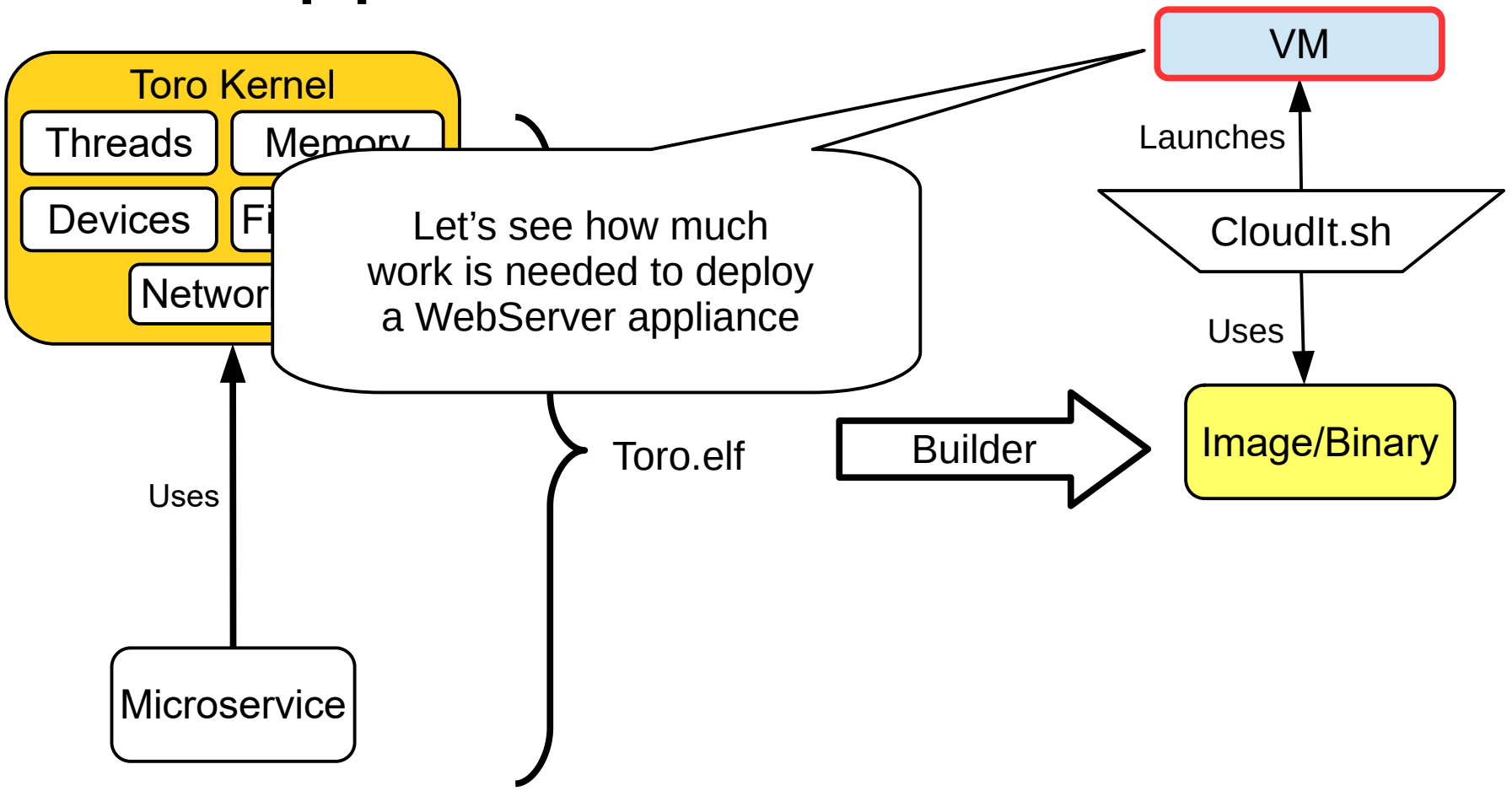
[1]“Immutable Infrastructure”, Stella

The generated image/binary is **Immutable[1]**, i.e., the generated image can be used across different hypervisors without the need to recompile it.

Application-oriented Kernel



Application-oriented Kernel

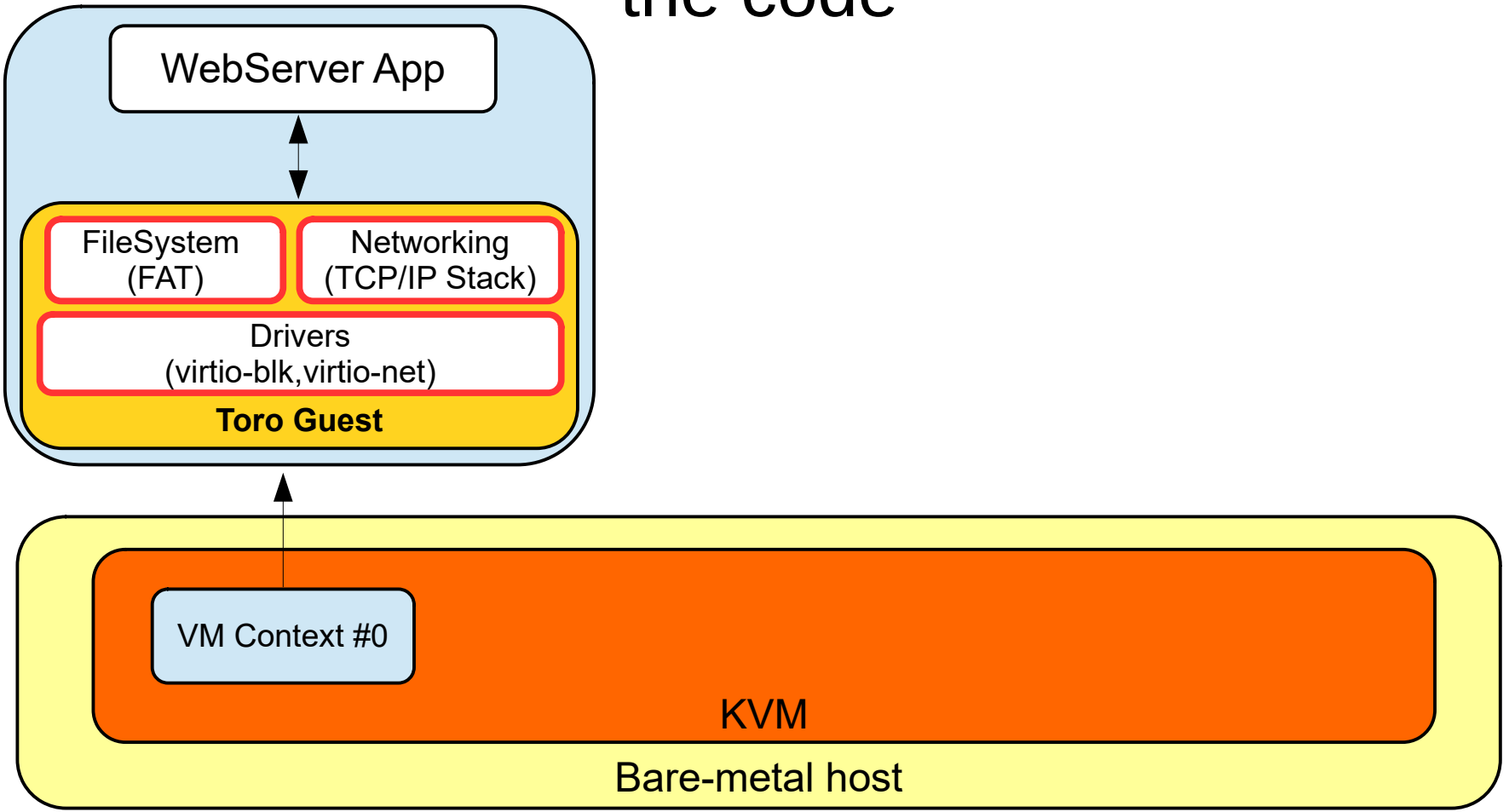


The WebServer Appliance

- Simple microservice that serves files by using the HTTP protocol
  - Find it at https://github.com/torokernel/torokernel among other examples
  - This appliance is used to host Toro's website (http://www.torokernel.io and click on "View on Toro")

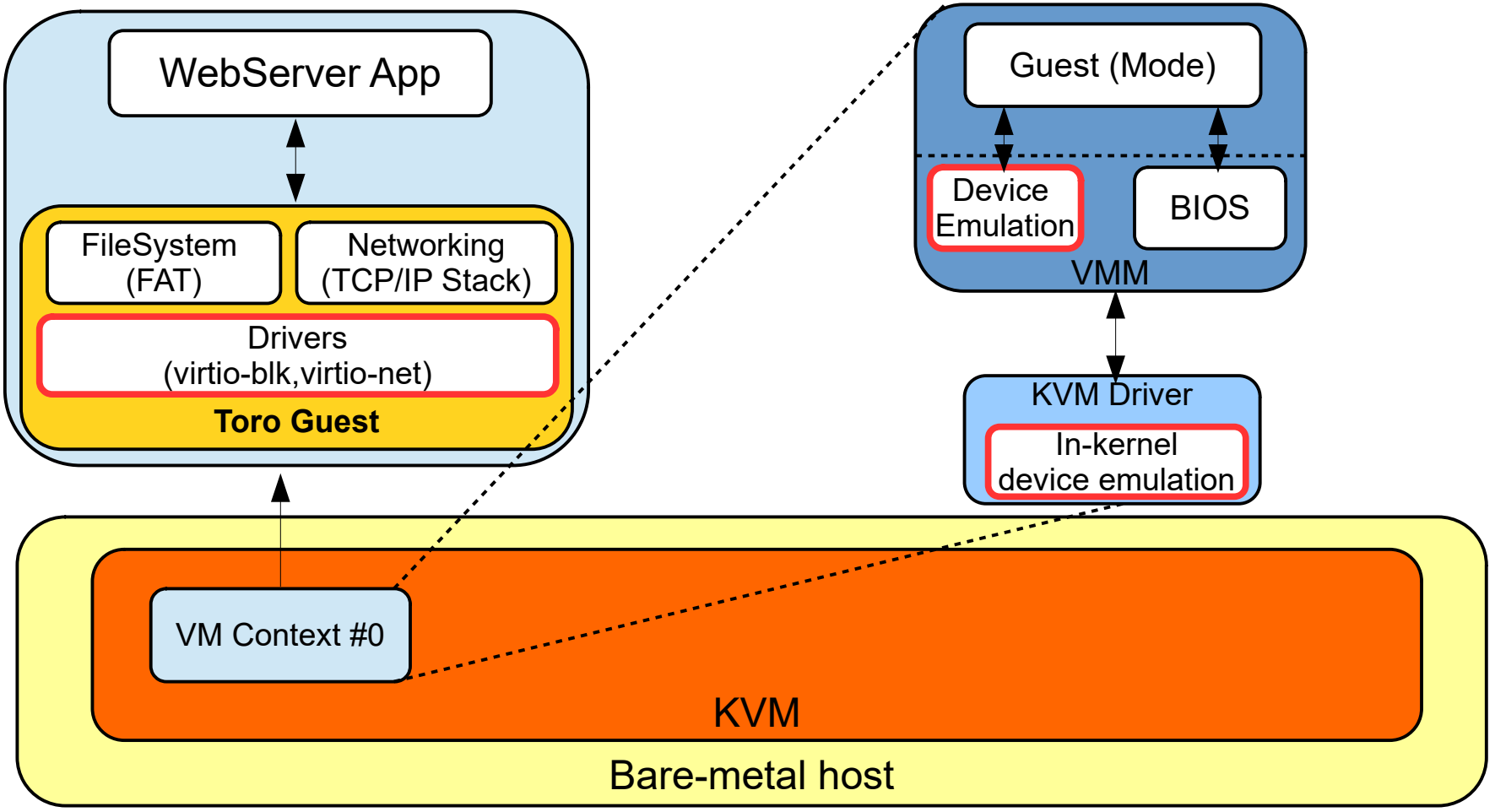
How the appliance is setup?

the code



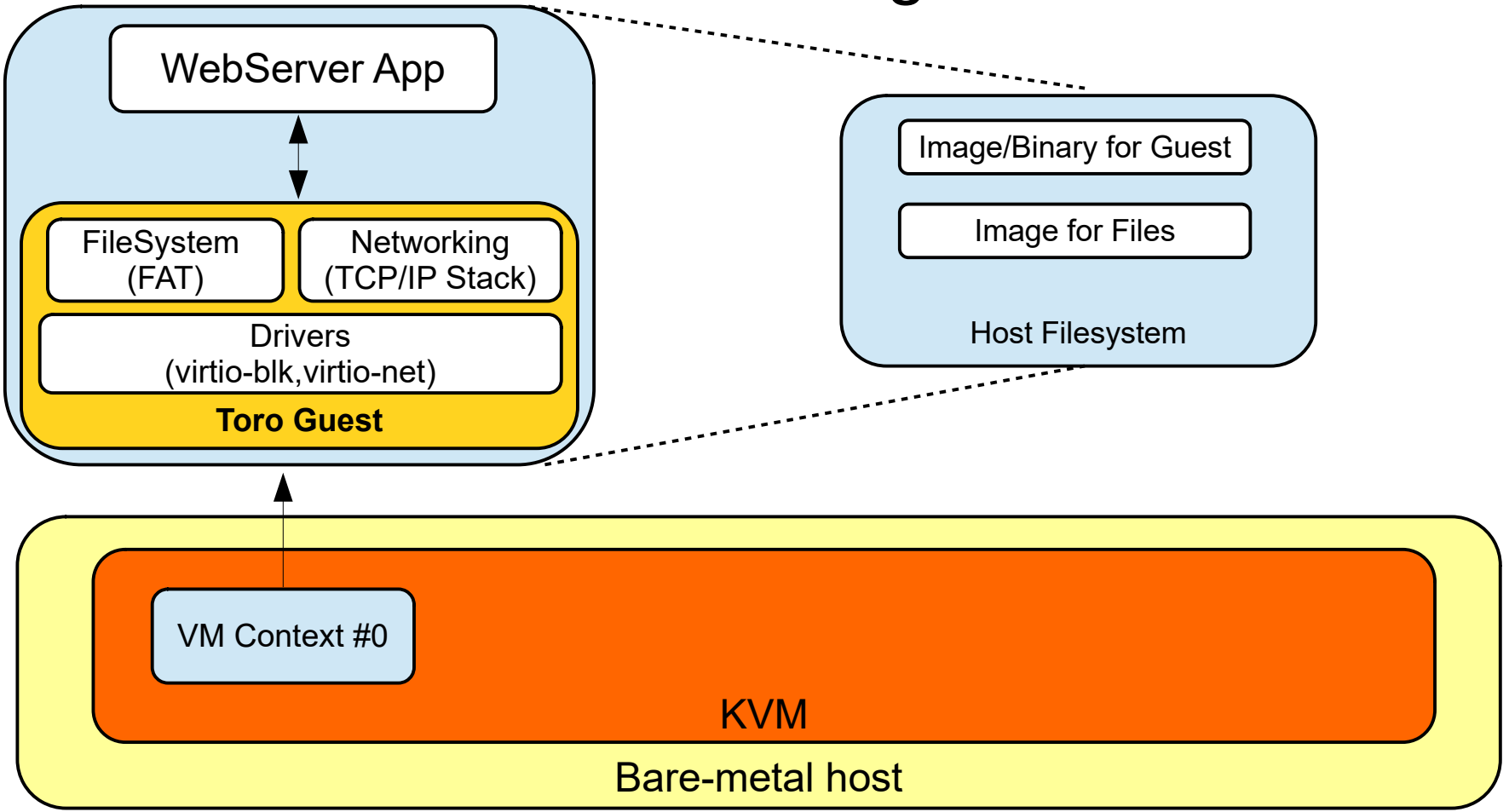
How the appliance is setup?

device model



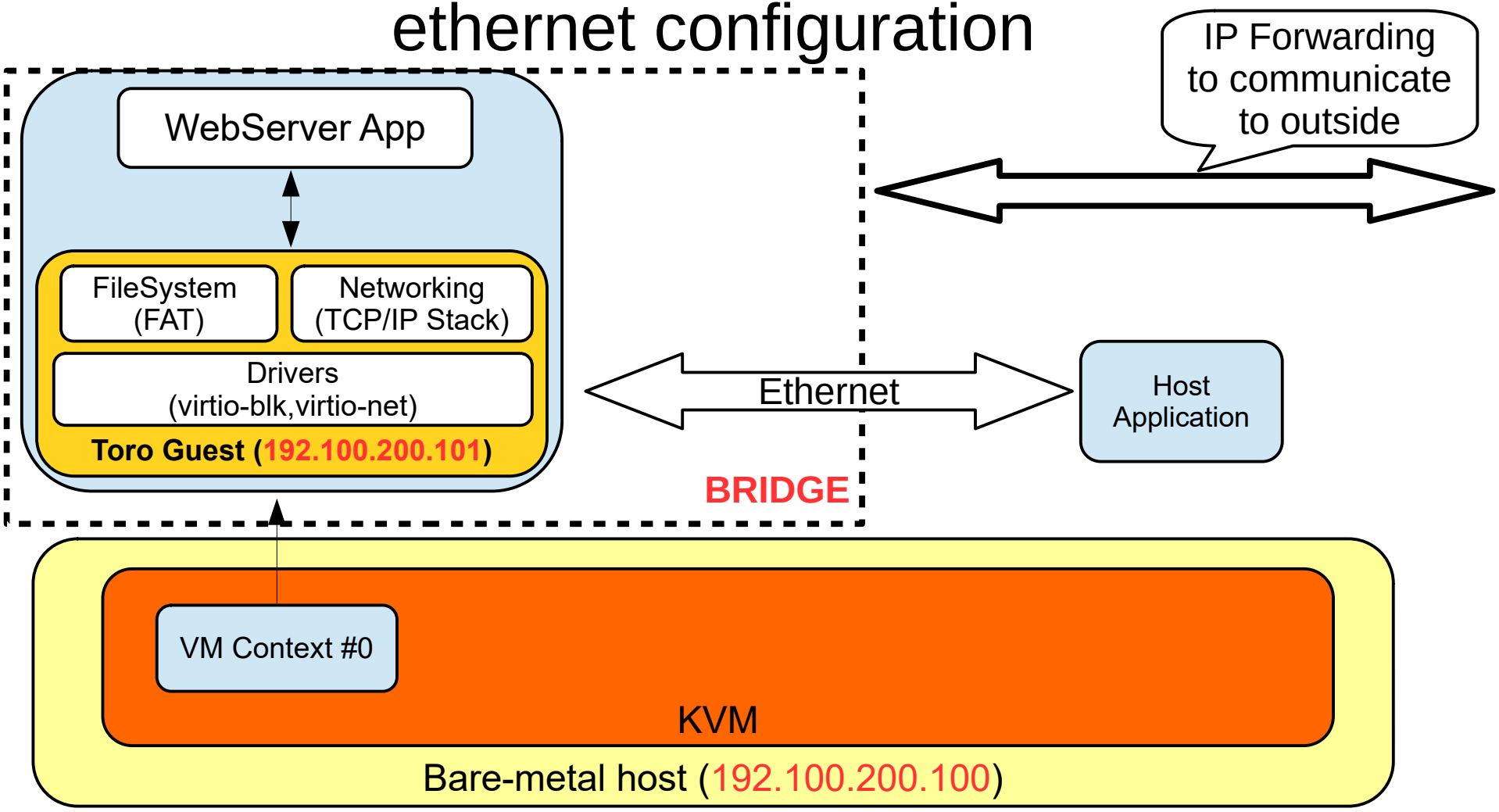
How the appliance is setup?

on-disk images



How the appliance is setup?

ethernet configuration

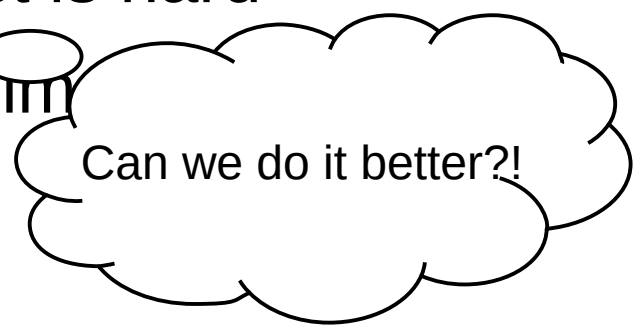


The Static WebServer
drawbacks

- Disk images consume memory and on-disk space, e.g., each guest has its own image
- The use of a TCP/IP stack requires configurations, e.g., bridge, an IP per guest, guest drivers, devices
- The use of more devices increases the attack surface
- Sharing of files between guests and host is hard
- Relying on a specific FS is not good for immutable images

The Static WebServer
drawbacks

- Disk images consume memory and on-disk space, e.g., each guest has its own image
- The use of a TCP/IP stack requires configurations, e.g., bridge, an IP per guest, guest drivers, devices
- The use of more devices increases the attack surface
- Sharing of files between guests and host is hard
- Relying on a specific FS is not good for m



The Static WebServer drawbacks

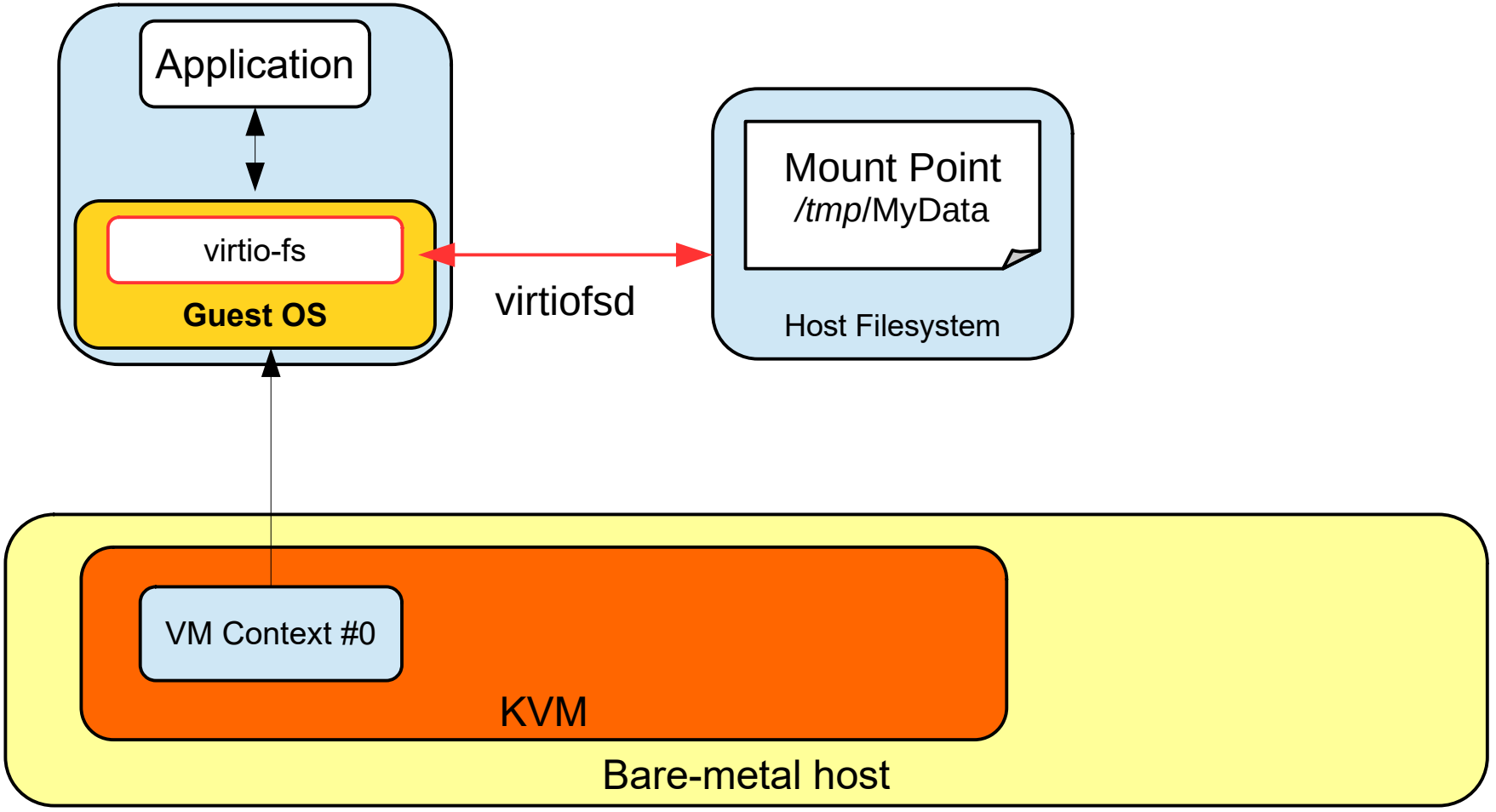
- Disk images consume memory and on-disk space, e.g., each guest has its own image
- The use of a TCP/IP stack requires configuration, e.g.

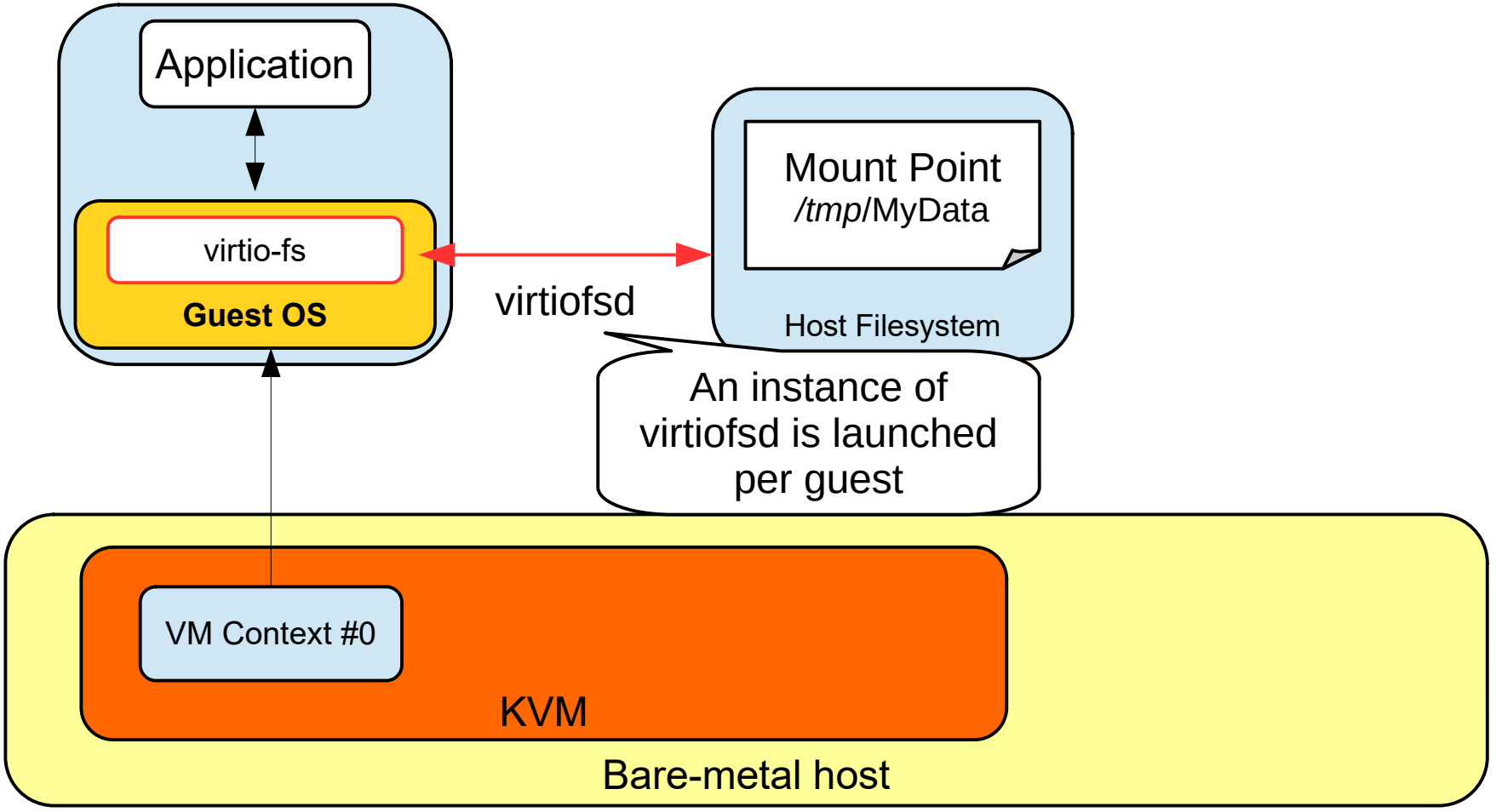
We propose to use of **virtio-fs** for filesystem and **virtio-vsocket** for networking to simplify toro unikernel code, reduce attack surface and ease appliance configuration

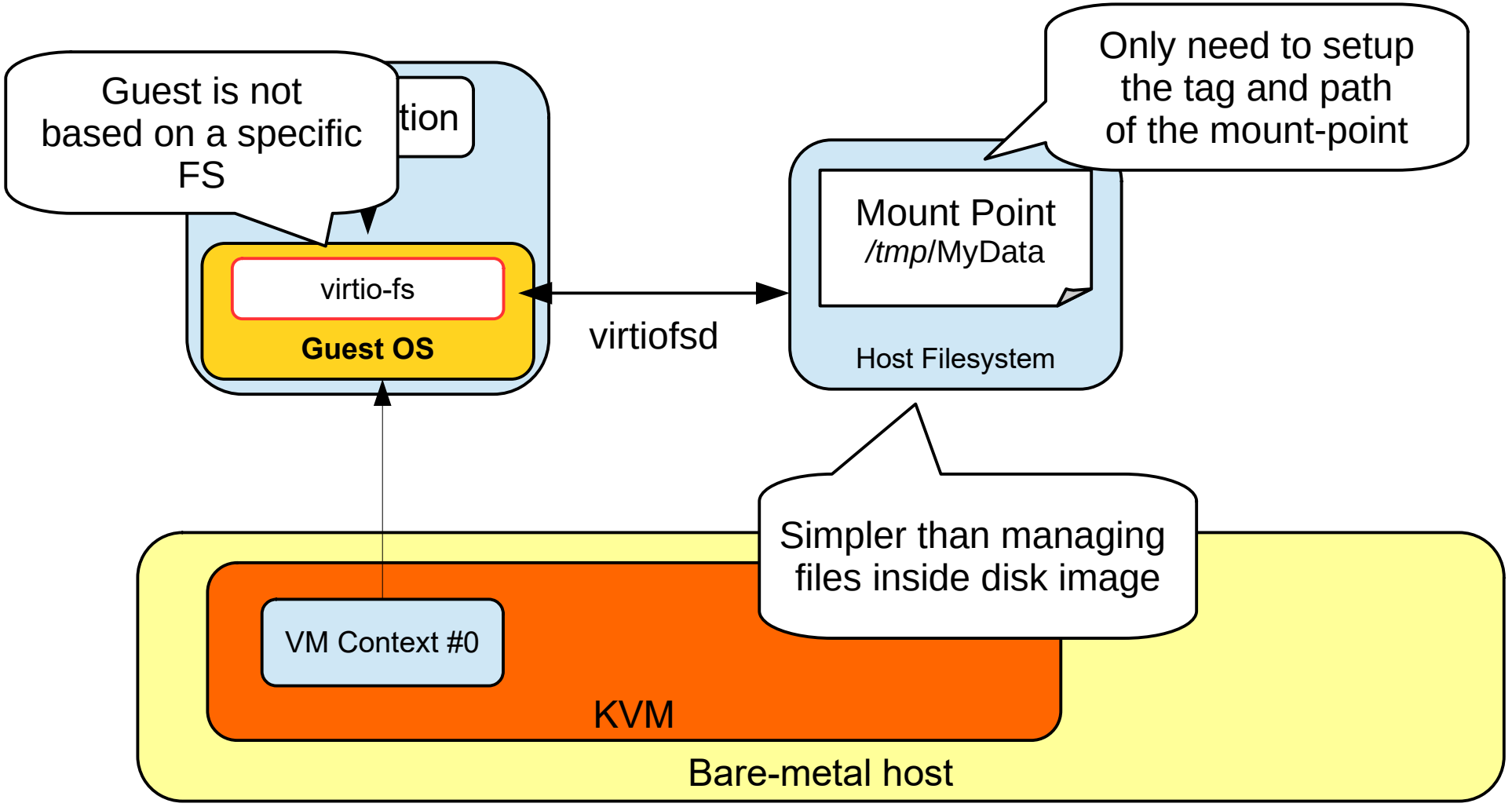
- Sharing of files between guests and host is hard
- Relying on a specific FS is not good for m

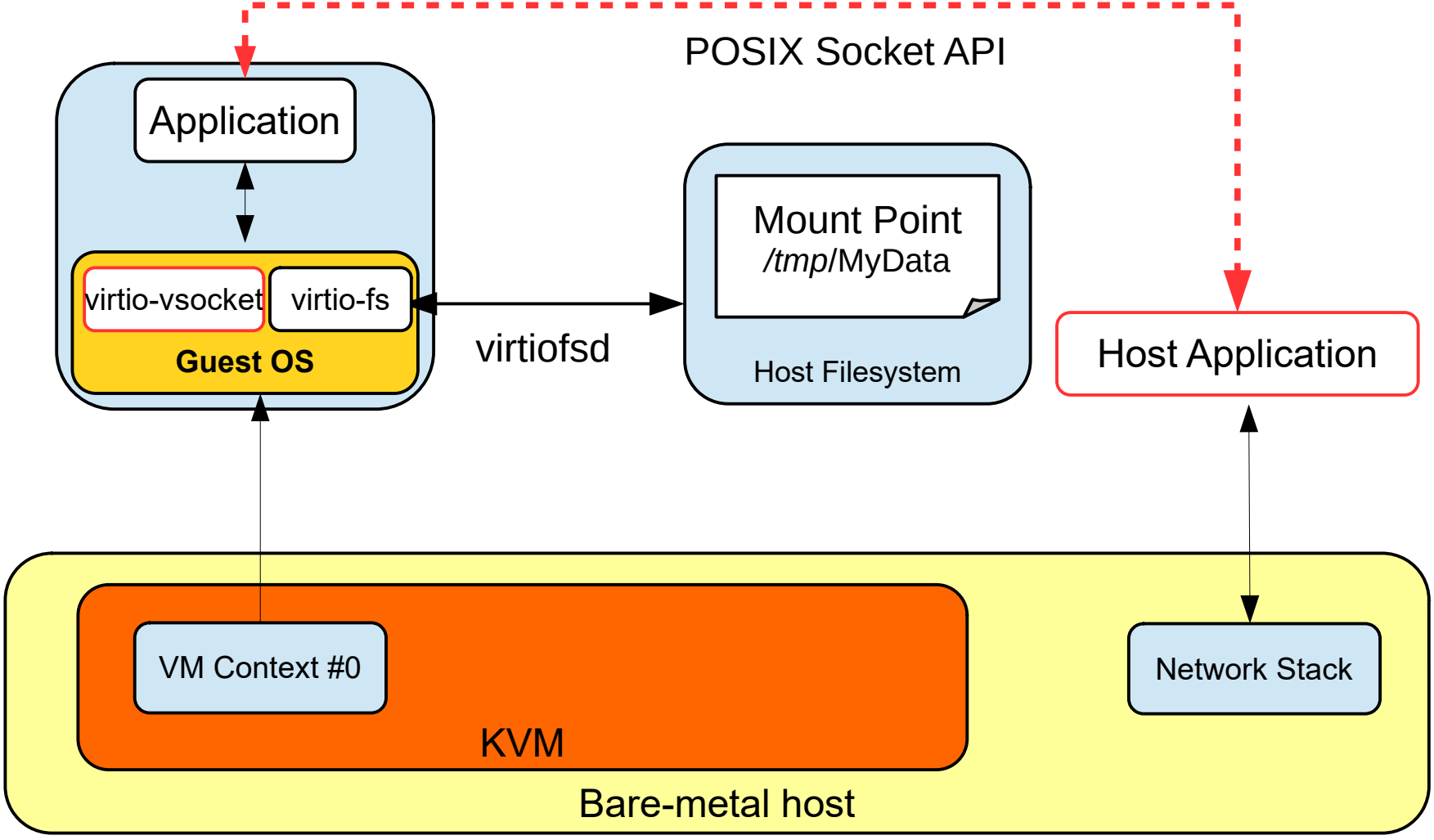
What are Virtio-fs and Virtio-vsocket?

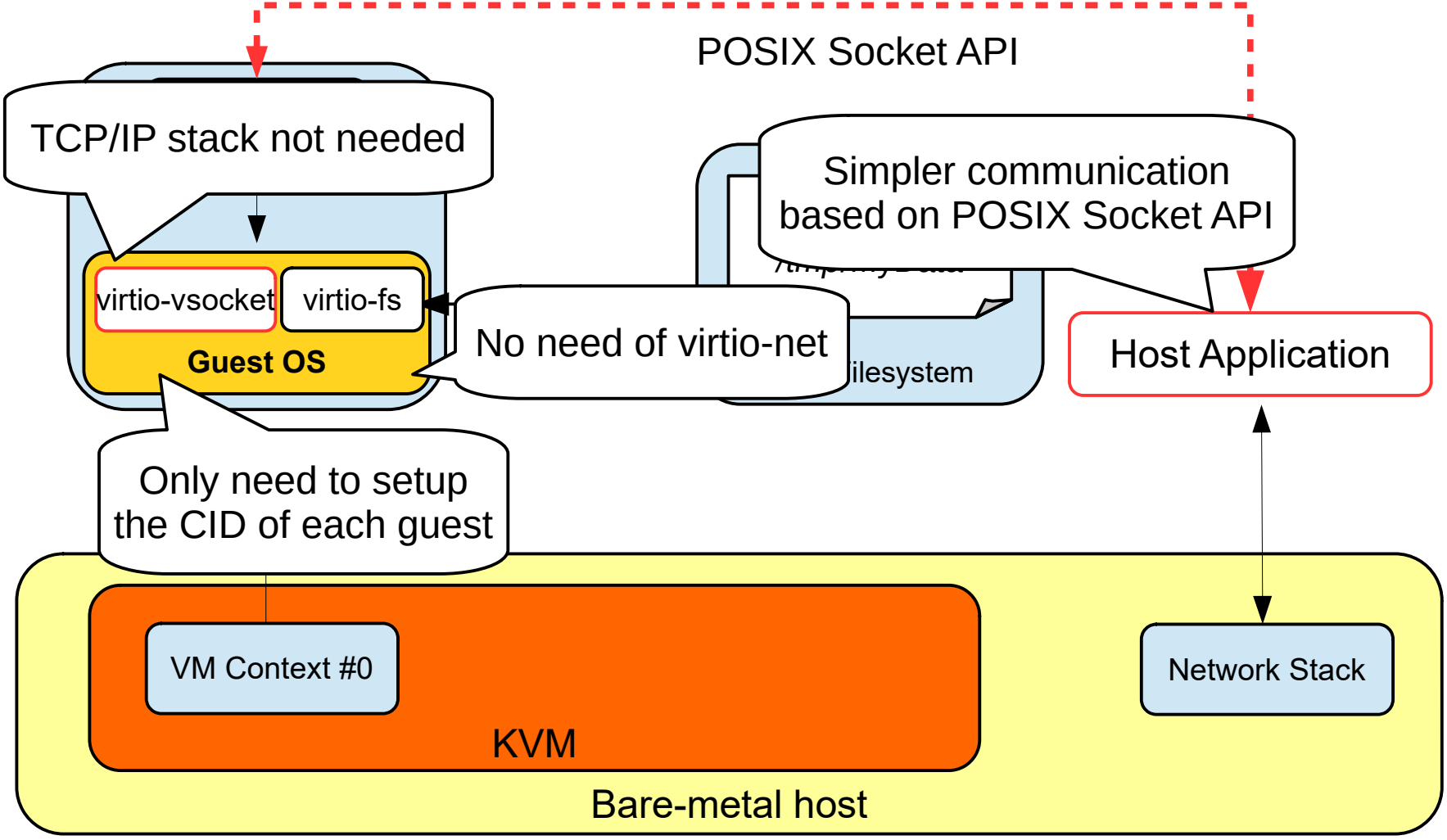
- Virtio devices
- Virtio-fs is a shared file system that allows VMs to access a directory tree on the host (more information at https://virtio-fs.gitlab.io/)
- Virtio-vsocket allows communication between guest and host by using POSIX Sockets API (more information at https://wiki.qemu.org/Features/VirtioVsock)


Application
virtio-fs
Guest OS
virtiofsd
Mount Point
/tmp/MyData
Host Filesystem
VM Context #0
KVM
Bare-metal host


Application
virtio-fs
Guest OS
virtiofsd
Mount Point
/tmp/MyData
Host Filesystem
An instance of virtiofsd is launched per guest
VM Context #0
KVM
Bare-metal host


Guest is not based on a specific FS
tion
virtio-fs
Guest OS
virtiofsd
Only need to setup the tag and path of the mount-point
Mount Point
/tmp/MyData
Host Filesystem
Simpler than managing files inside disk image
VM Context #0
KVM
Bare-metal host


POSIX Socket API
Application
virtio-vsocket
virtio-fs
Guest OS
virtiofsd
Mount Point
/tmp/MyData
Host Filesystem
Host Application
VM Context #0
KVM
Network Stack
Bare-metal host


POSIX Socket API
TCP/IP stack not needed
Simpler communication
based on POSIX Socket API
virtio-vsocket
virtio-fs
Guest OS
No need of virtio-net
ilesystem
Host Application
Only need to setup
the CID of each guest
VM Context #0
KVM
Network Stack
Bare-metal host

Results

- Reduce Networking source code
  - -1126 LoC → ~ 48% LoC less in Networking
- User appliance has no change
- Reduce FileSystem source code
  - -1046 LoC → ~ 54% LoC less in FileSytem
- Simplified appliance deployment
  - Launch virtiofsd and set the CID per instance
- Communication outside host is provided by an user application named VSocketProxy
  - This is ongoing work
- Currently deployed at http://www.torokernel.io and then go to “View on Toro”

Demo

Future Work

- Microvm
- Improve drivers, e.g., zero-copy in virtio-vsocket
- Formal specification of VirtIO
- Commit VSocketProxy

QA

Use Case: WebServer