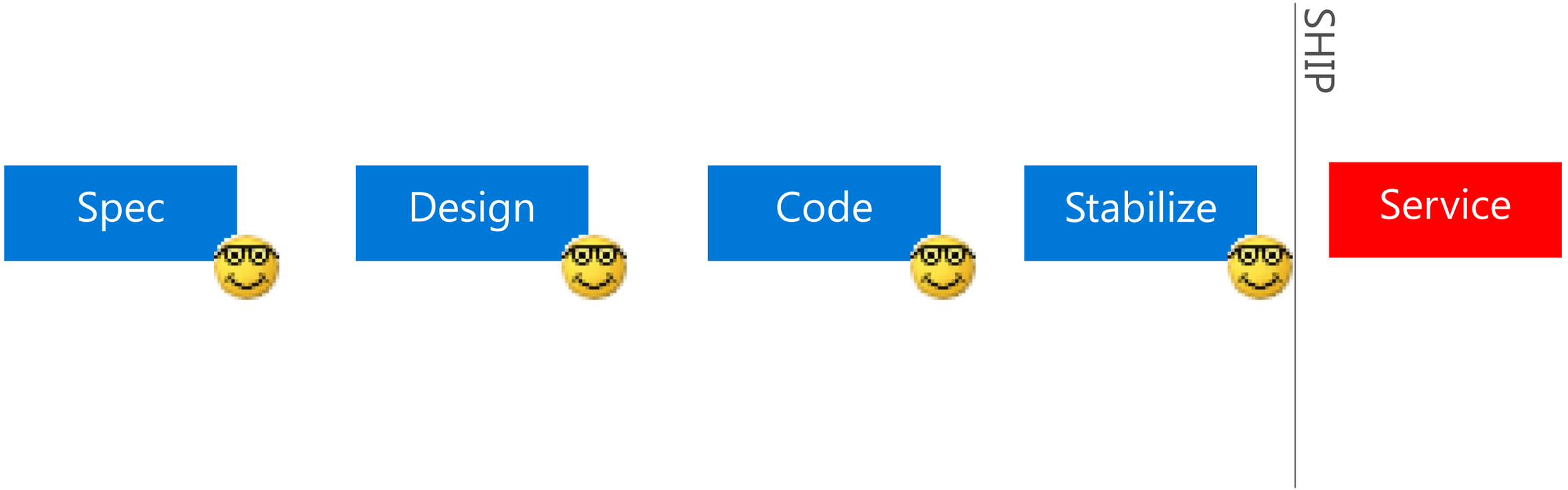
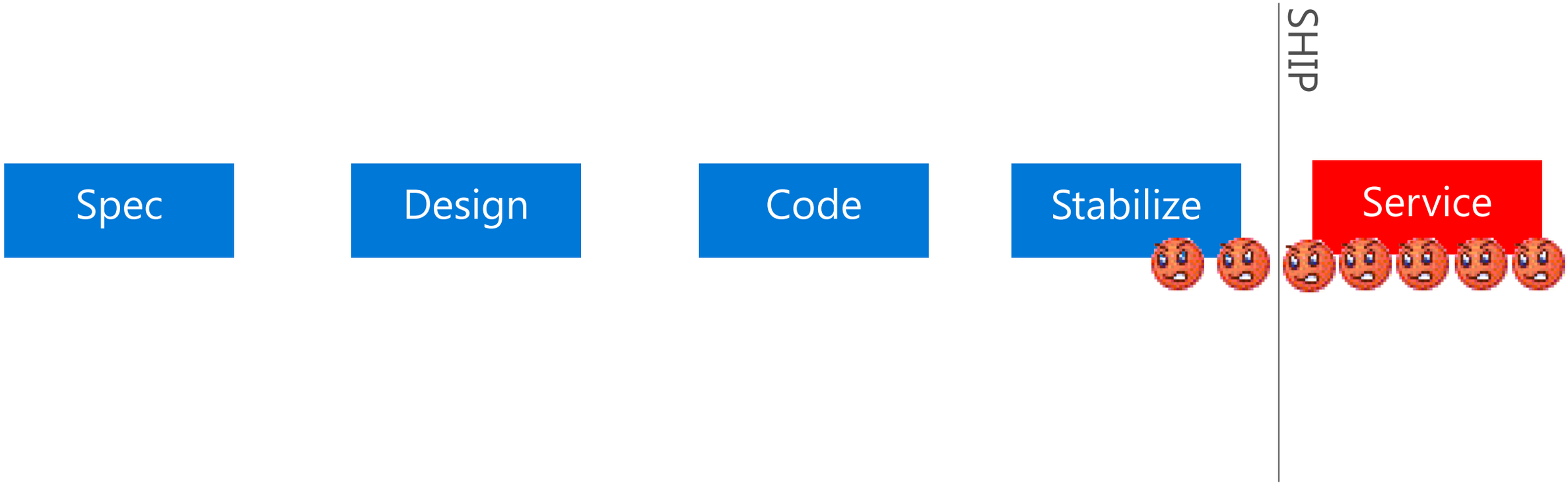
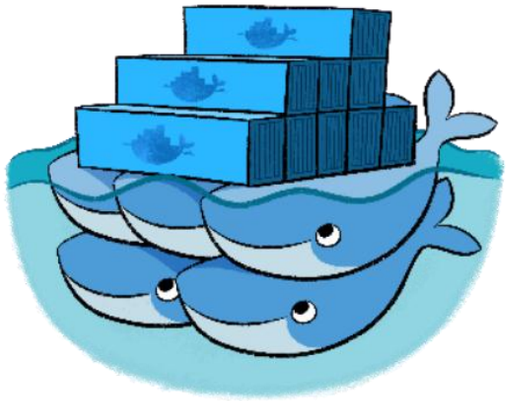
Breaking Things Early: Designing Secure Containers

Saruhan Karademir
Security Software Engineer
WDG Security Assurance


Spec
Design
Code
Stabilize
SHIP
Service

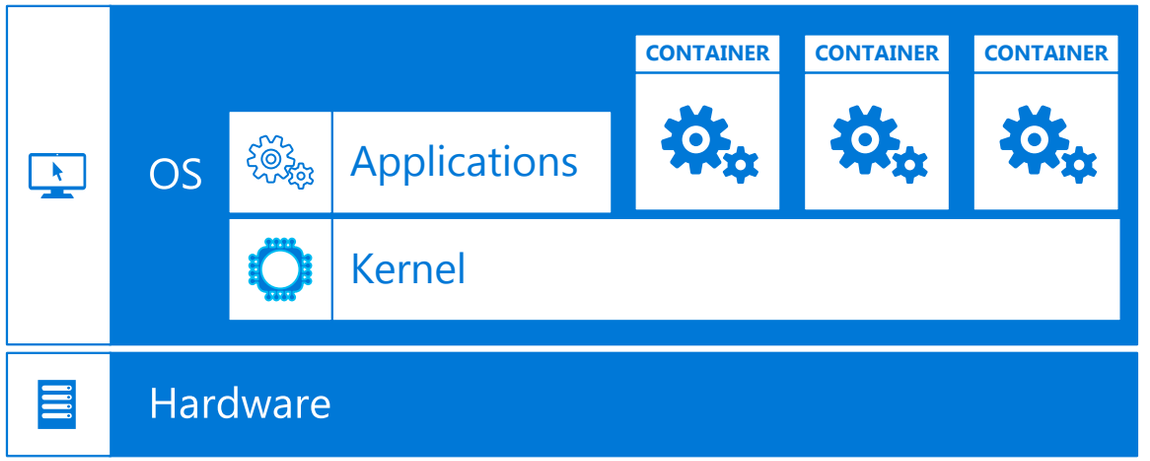

Spec
Design
Code
Stabilize
Service
SHIP

Containers

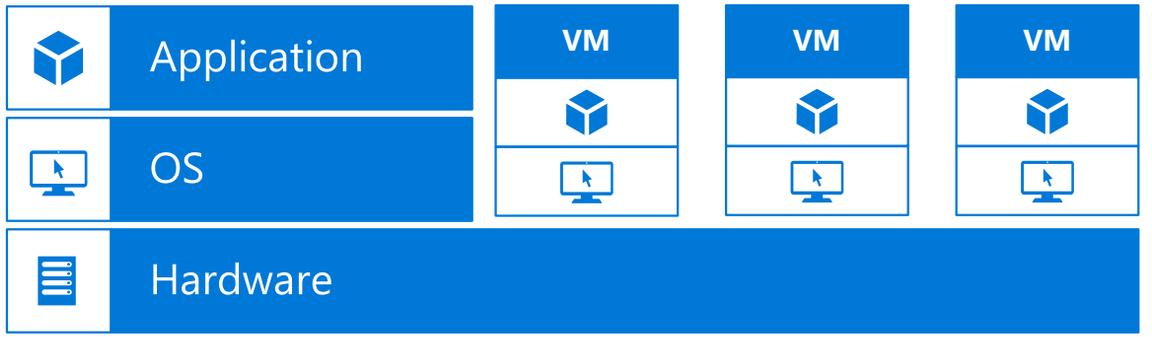


What is a container?

Containers = Operating system virtualization

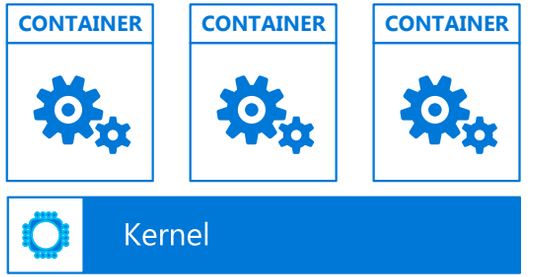


Traditional virtual machines = hardware virtualization



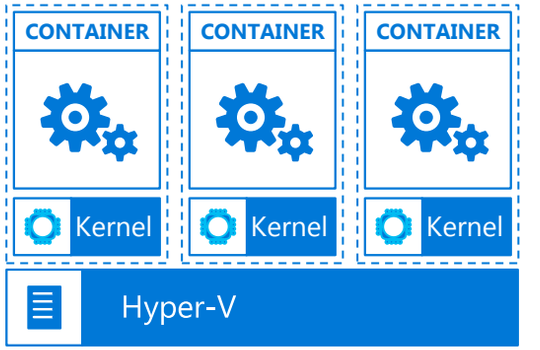
Windows Server Containers

Maximum speed and density

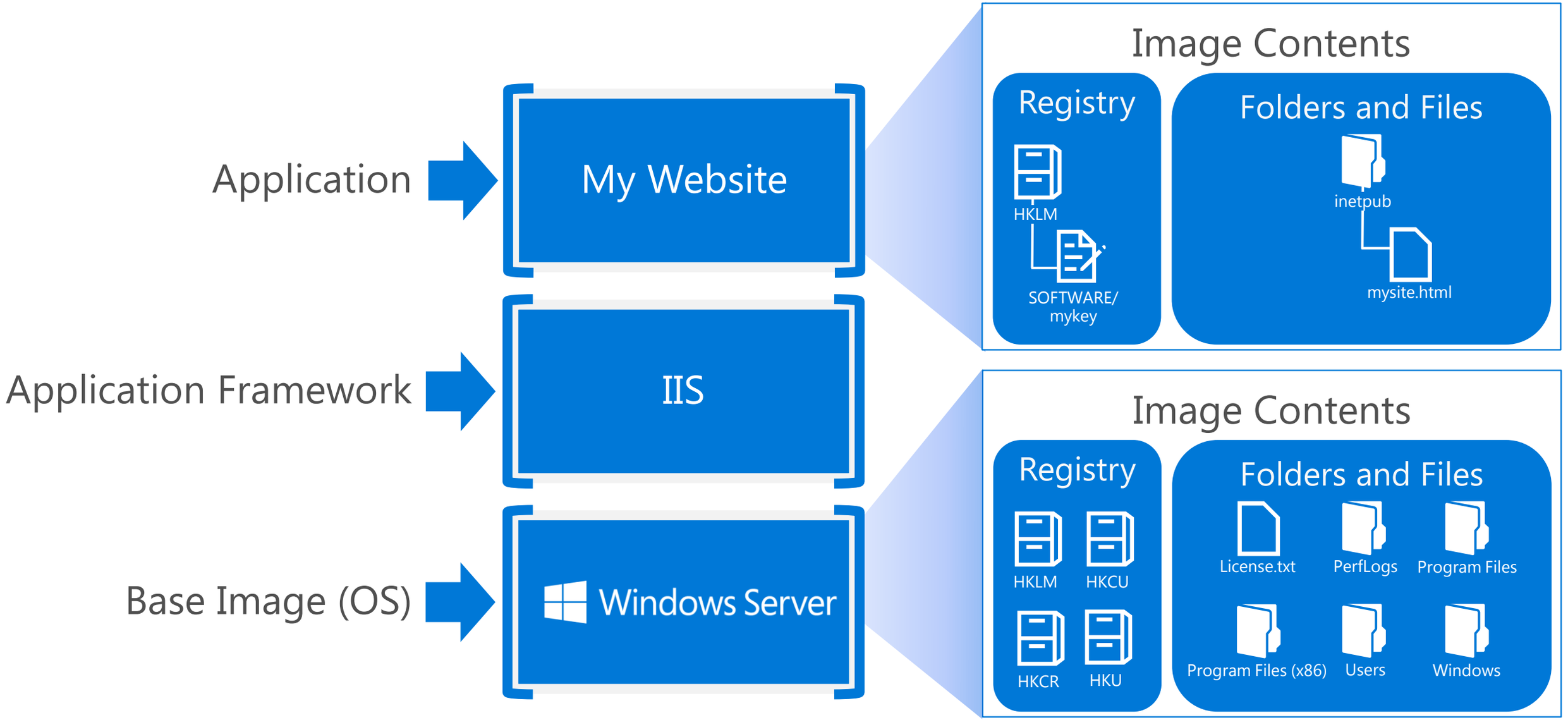


Hyper-V Containers

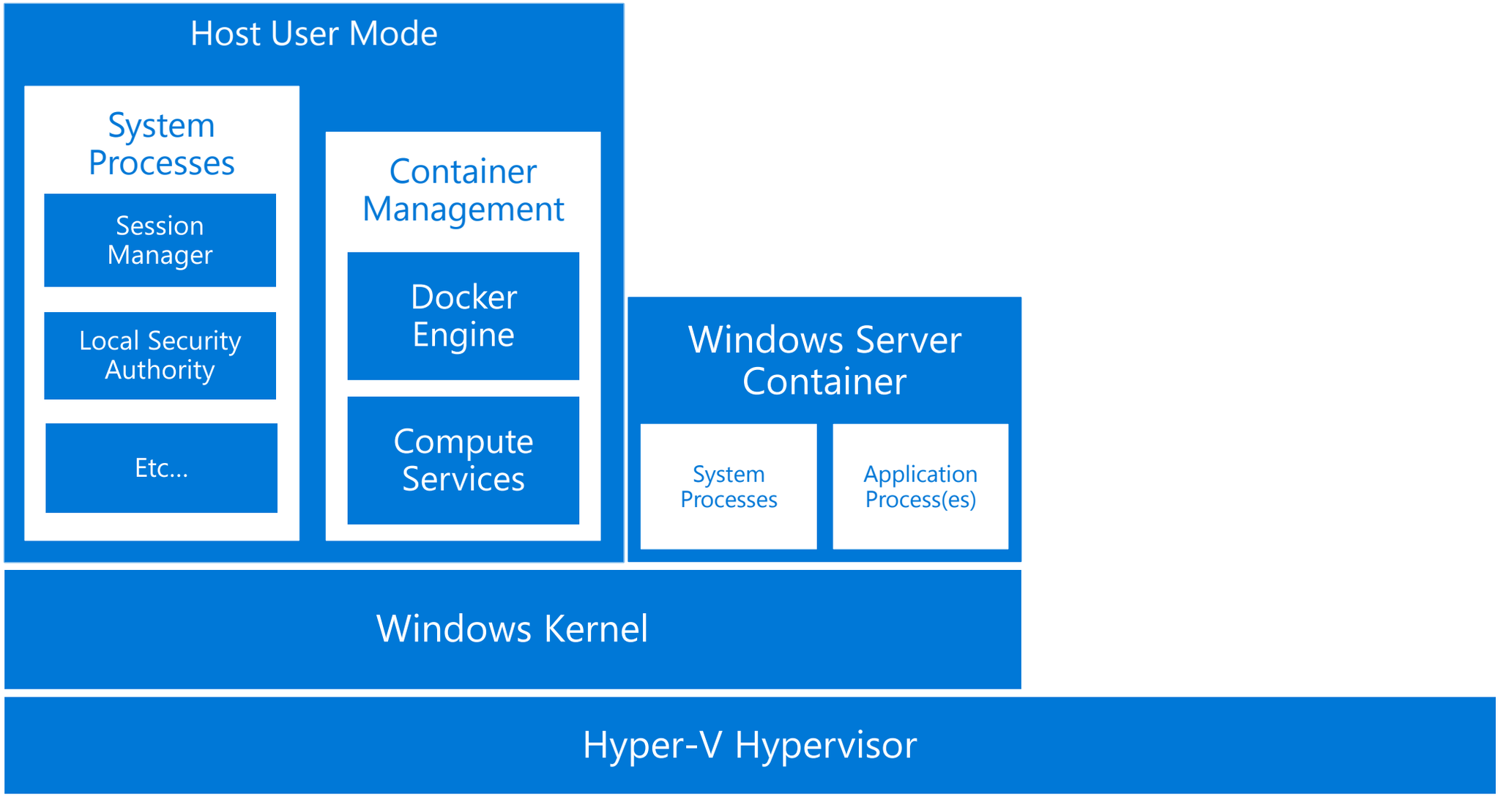
Isolation plus performance



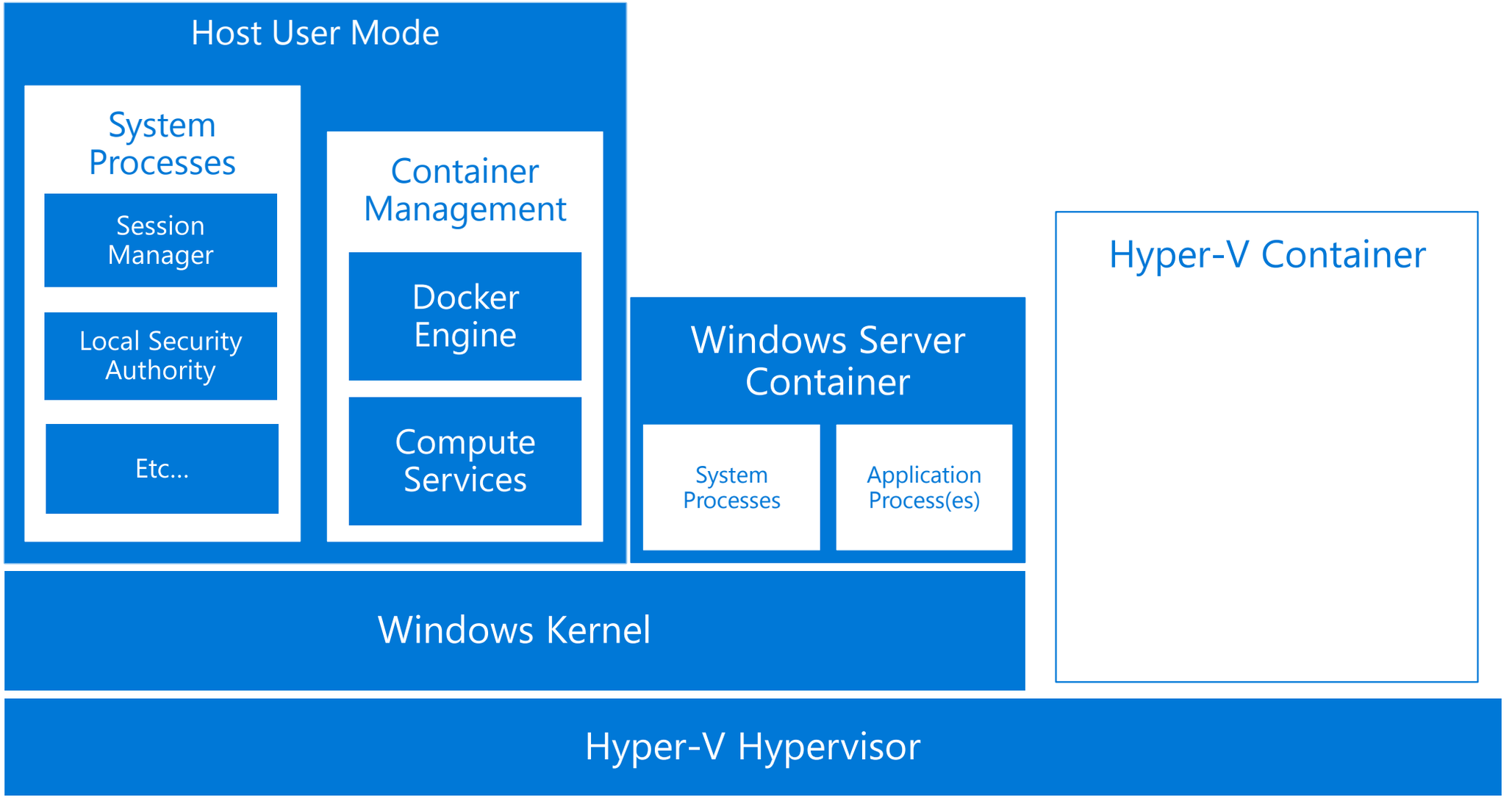
Layered File System



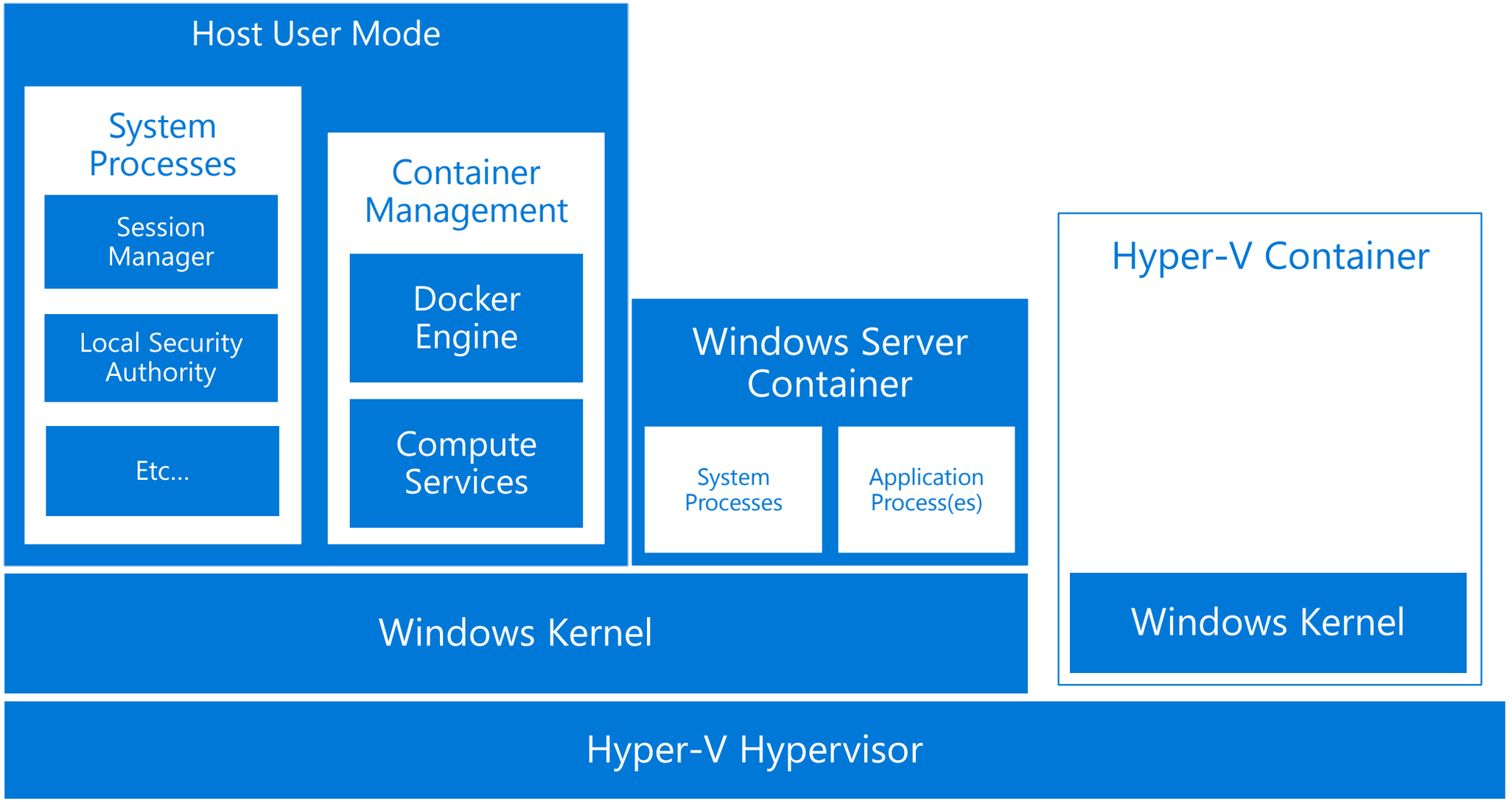
Windows Containers



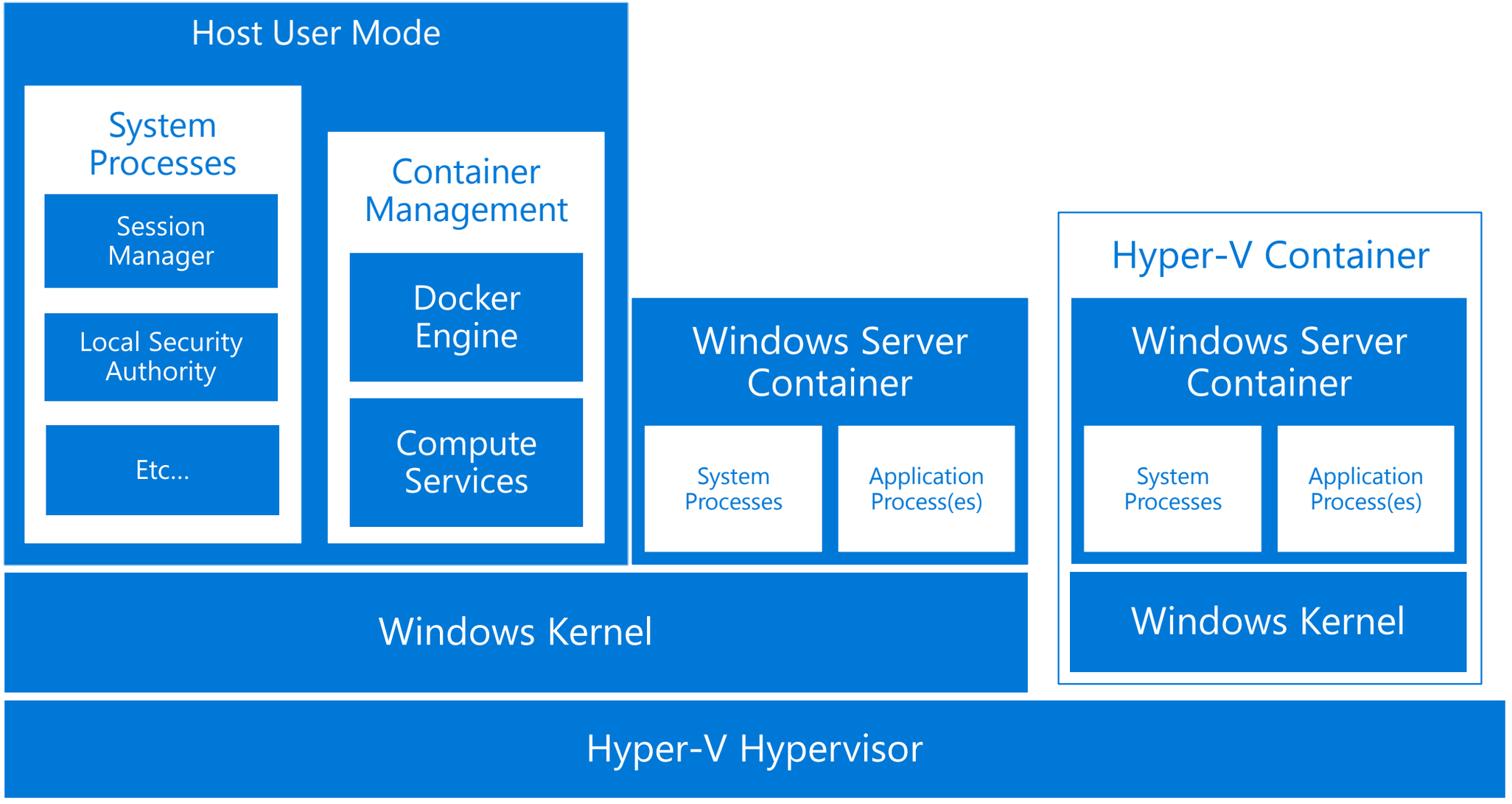
Windows Containers



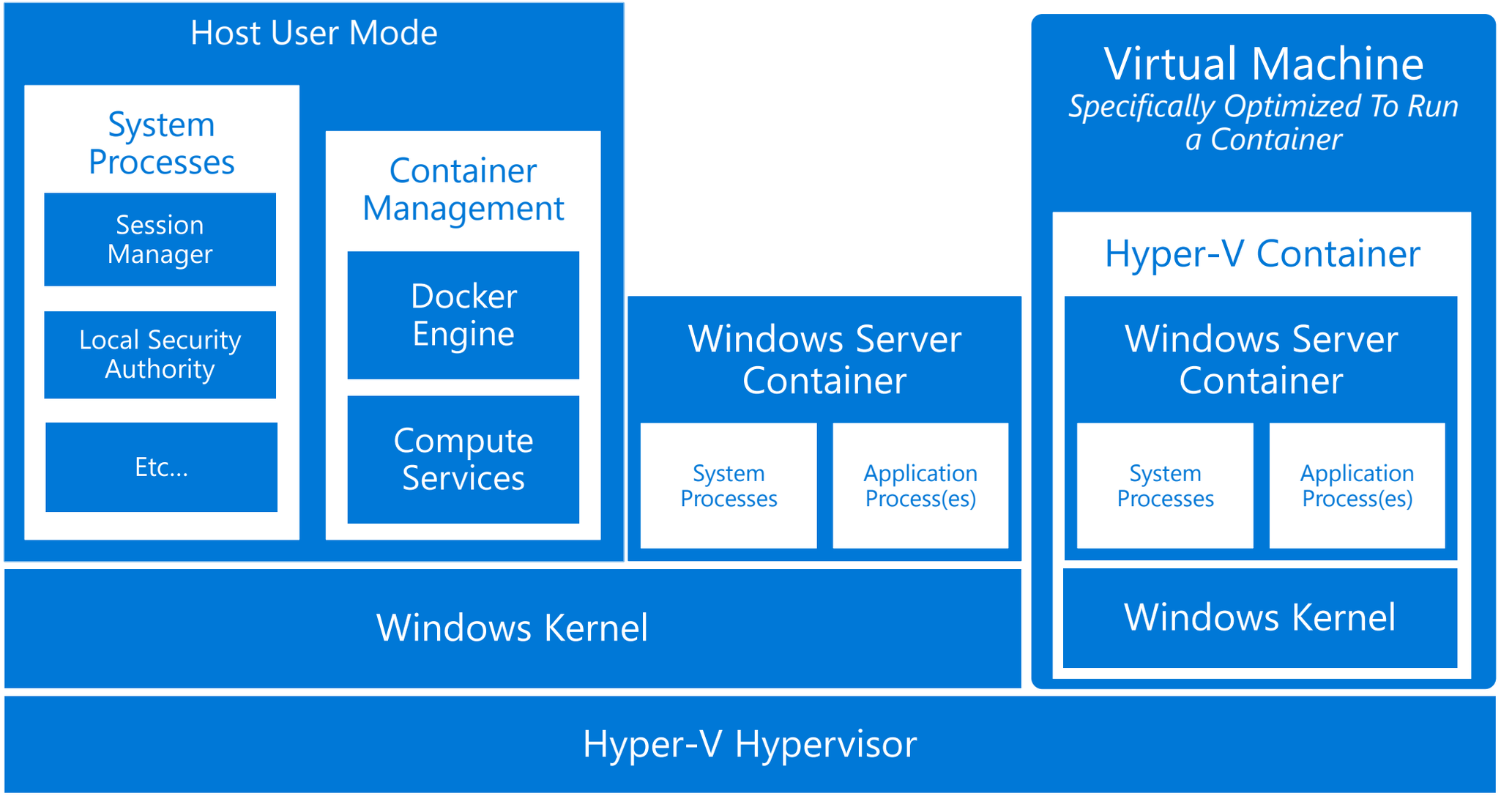
Windows Containers



Windows Containers



Windows Containers



Three Stories

Story 1 – Golang to Native

Story 2 – Container Identity

Spec

Design

Code

Stabilize

SHIP

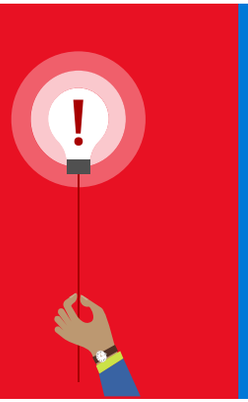
Service

Story 3 – Storage Architecture

Story 1: Safe(r) golang<->C bridge

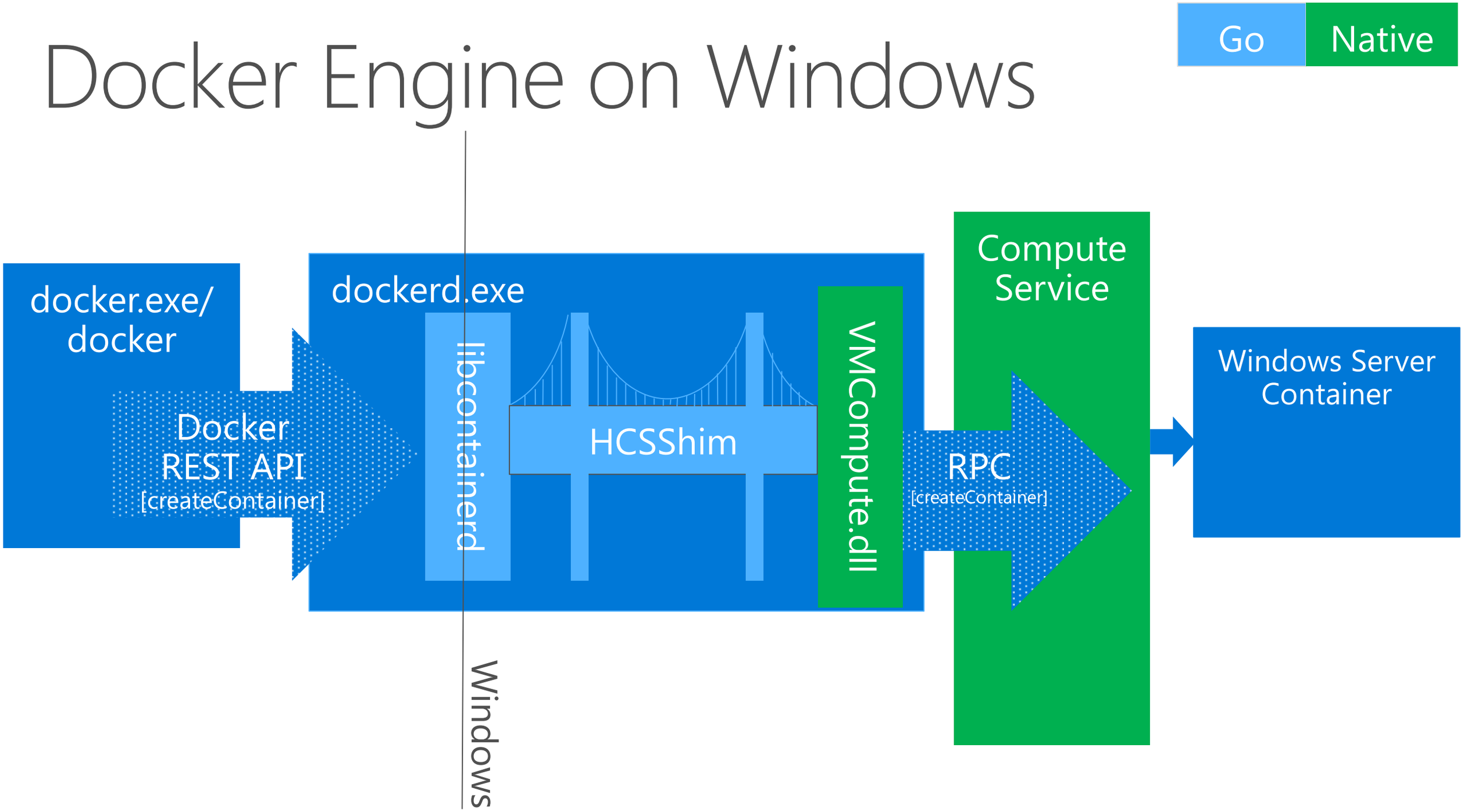
You are here

Spec → Design → Code → Stabilize → SHIP → Service



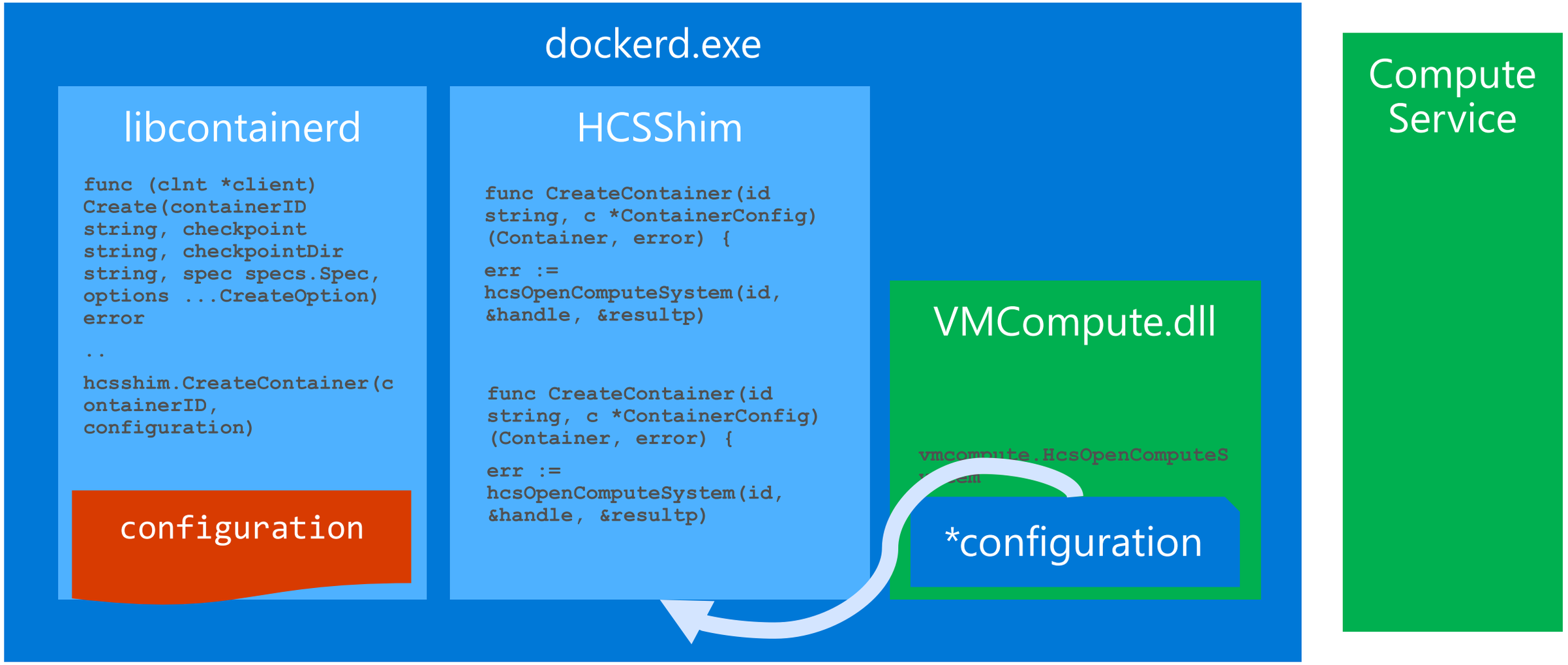
Variables passed from Go to C may be garbage collected early.

Docker Engine on Windows



Diving into Docker Engine

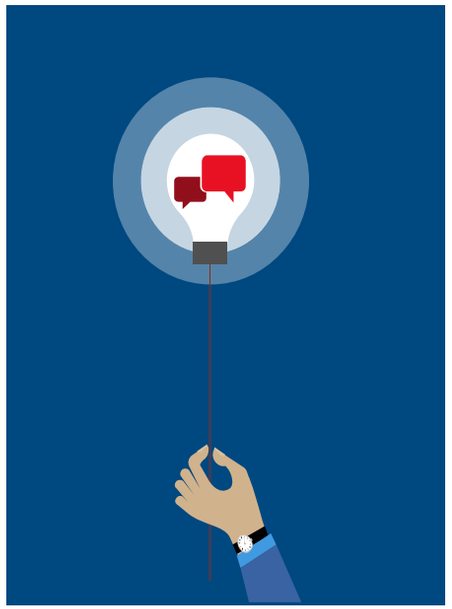
Go | Native



```
132 -       // Call the procedure itself.
133 -       r1, _, _ := proc.Call(
134 -               uintptr(unsafe.Pointer(idp)),
135 -               uintptr(unsafe.Pointer(paramsJsonp)),
136 -               uintptr(unsafe.Pointer(pid)),
137 -               stdinParam,
138 -               stdoutParam,
139 -               stderrParam)
140 -
141 -       use(unsafe.Pointer(idp))
142 -       use(unsafe.Pointer(paramsJsonp))
143 -
144 -       if r1 != 0 {
145 -               err = fmt.Errorf(title+" - Win32 API call returned error r1=%d err=%s id=%s params=%v", r1, syscall.Errno(r1), id, params)
```

```
99  +       err = createProcessWithStdHandlesInComputeSystem(id, string(paramsJson), &pid, stdinParam, stdoutParam, stderrParam)
100 +       if err != nil {
101 +               err := makeErrorf(err, title, "id=%s params=%v", id, params)
```

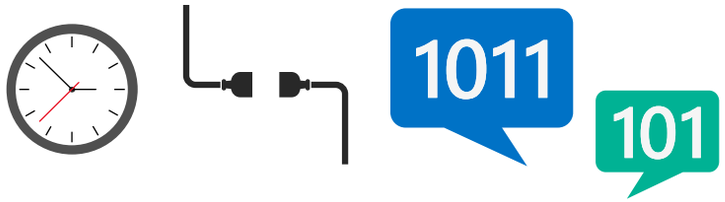
```
402 func createProcessWithStdHandlesInComputeSystem(id string, paramsJson string, pid *uint32, stdin *syscall.Handle, stdout *syscall.Handle, s
403         var _p0 *uint16
404         _p0, hr = syscall.UTF16PtrFromString(id)
405         if hr != nil {
406                 return
407         }
408         var _p1 *uint16
409         _p1, hr = syscall.UTF16PtrFromString(paramsJson)
410         if hr != nil {
411                 return
412         }
413         return _createProcessWithStdHandlesInComputeSystem(_p0, _p1, pid, stdin, stdout, stderr)
414 }
415
416 func _createProcessWithStdHandlesInComputeSystem(id *uint16, paramsJson *uint16, pid *uint32, stdin *syscall.Handle, stdout *syscall.Handle
417         if _perr := procCreateProcessWithStdHandlesInComputeSystem.Find(); _perr != nil {
418                 return _perr
419         }
420
421         r0, _, _ := syscall.Syscall6(procCreateProcessWithStdHandlesInComputeSystem.Addr(), 6, uintptr(unsafe.Pointer(id)), uintptr(unsafe.
422         if r0 != 0 {
423                 hr = syscall.Errno(r0)
424         }
```



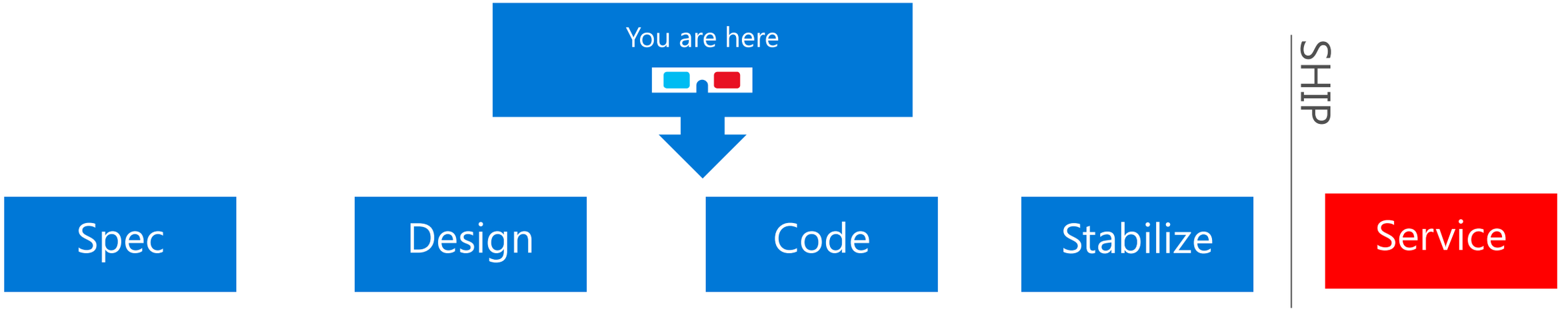
1. Pin the parameters sent to VMCompute.dll.

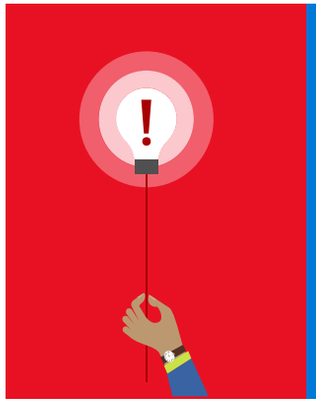
2. Follow the golang convention for pinning data.

3. Put the syscall in the function return, so variables are not GCed.



Story 2: Container Identity

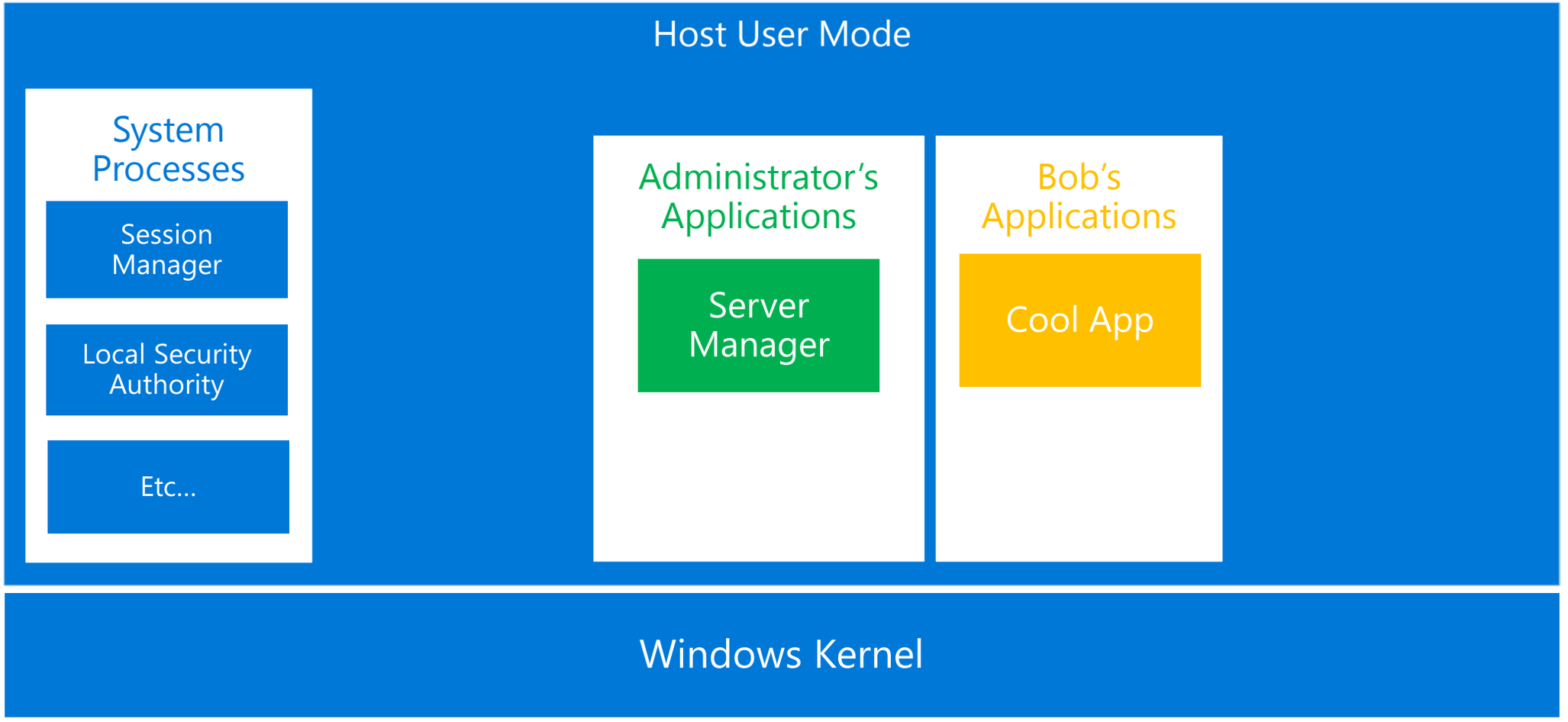




Processes inside a Windows Server Container ran at SYSTEM privileges.

Before Containers

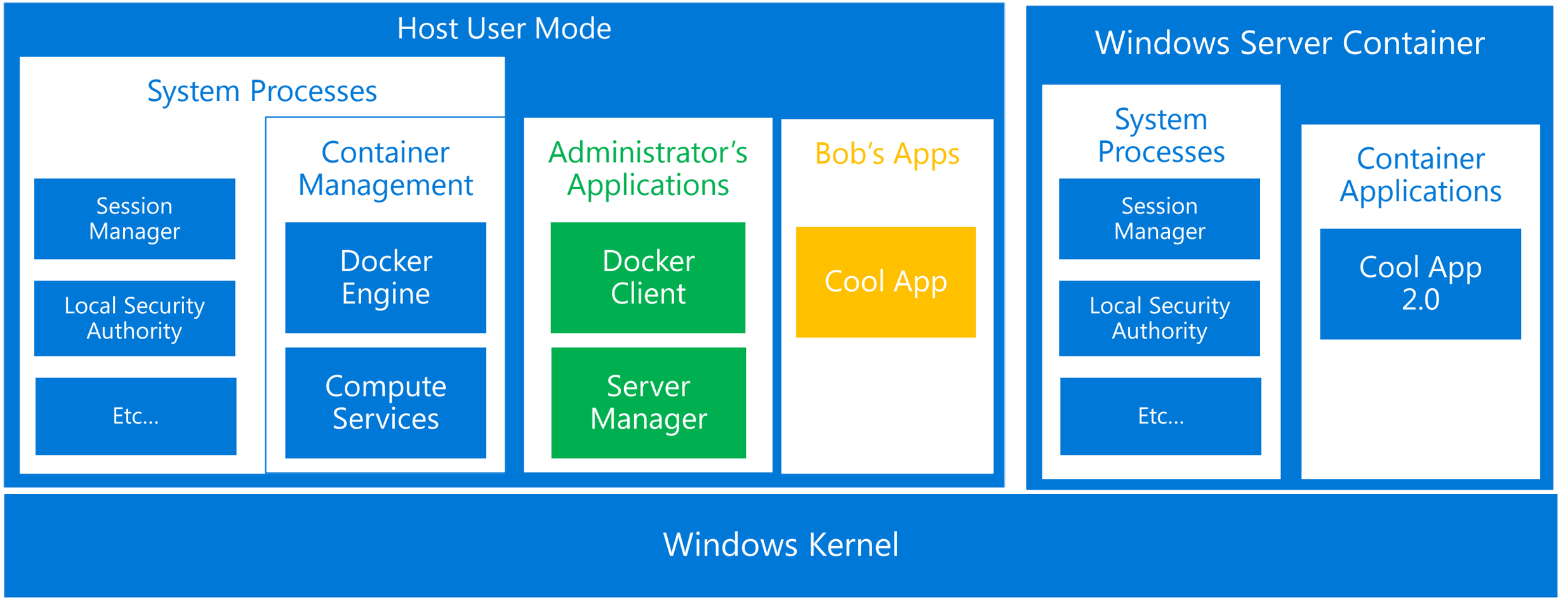
System | Admin | Bob



Then who was Container?

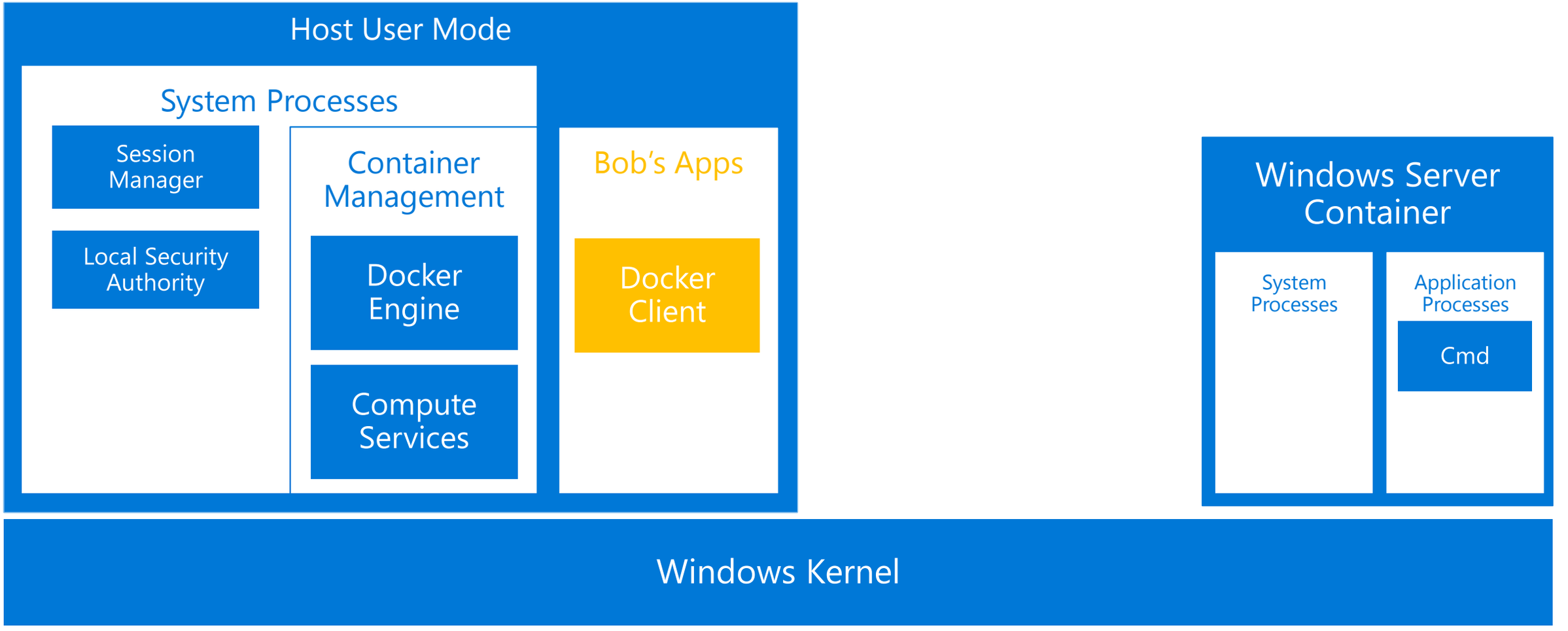
System | Admin | Bob

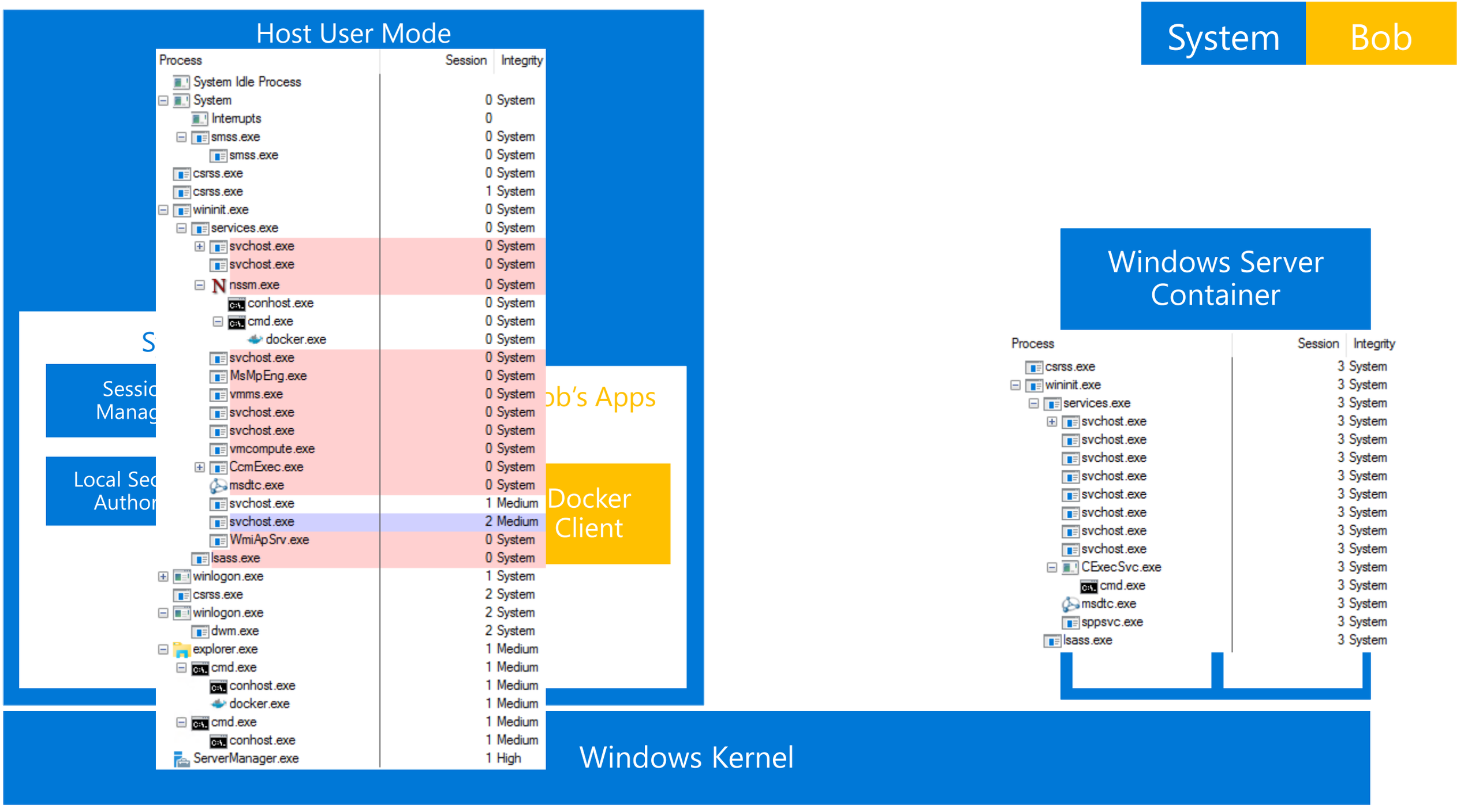
Containers do not share host's users



```
C:\Users\bob\>docker run -it
windowsservercore cmd
```

System | Bob



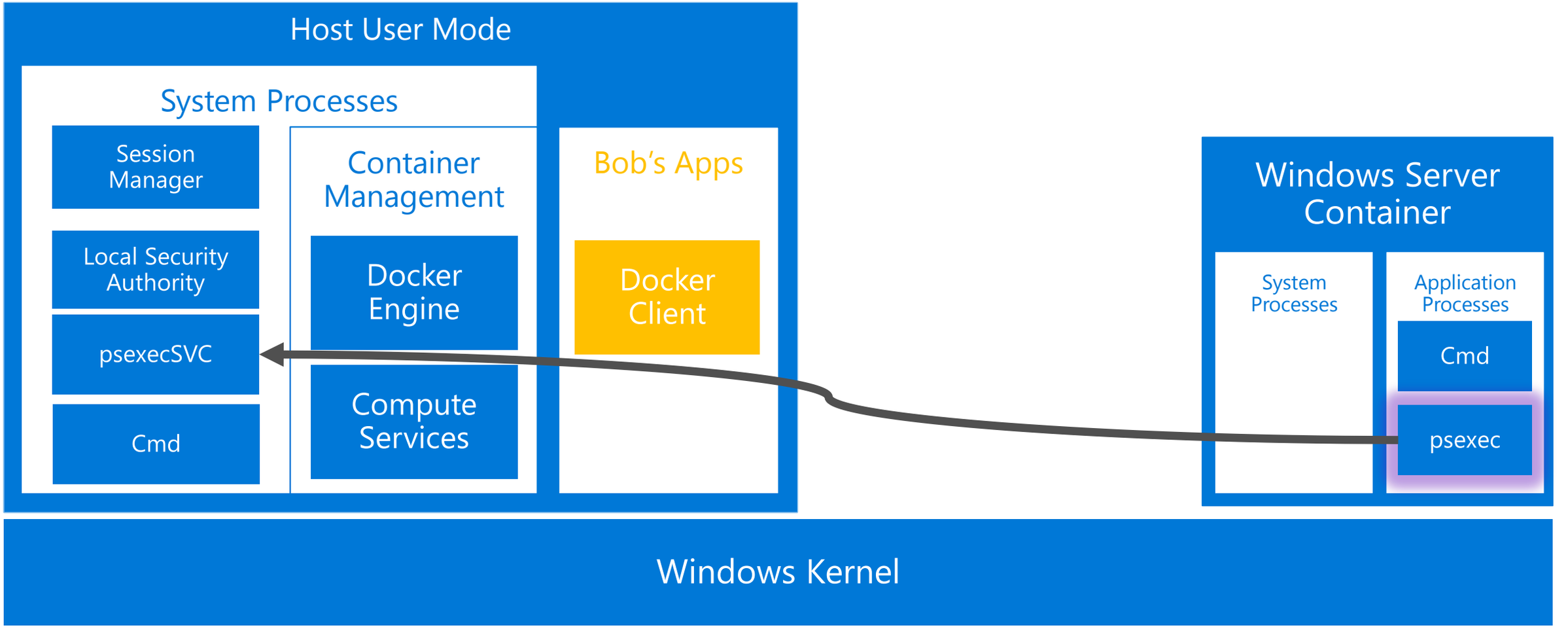


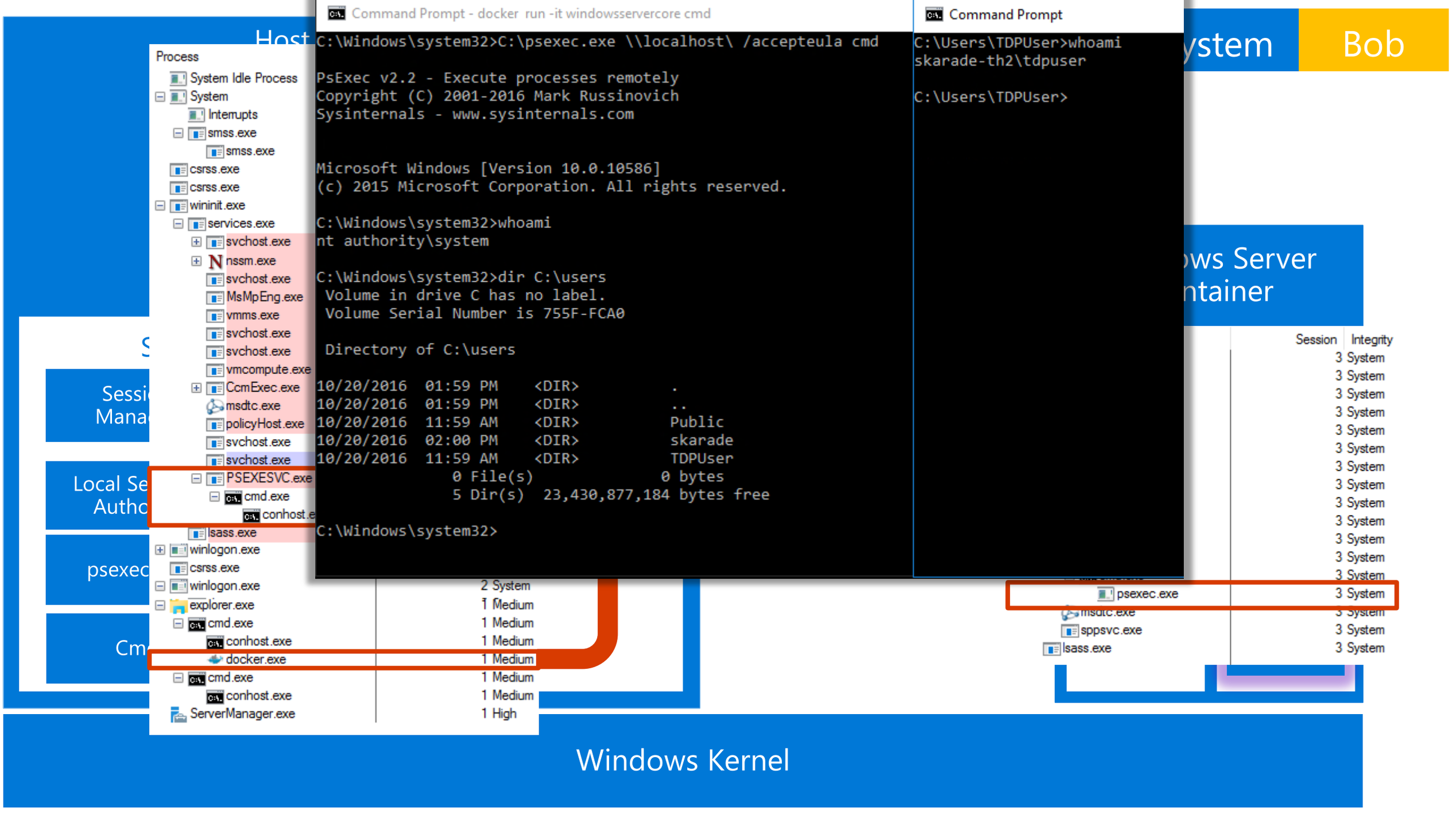
Process	Session	Integrity
System Idle Process		
System	0	System
Interrupts	0	
smss.exe	0	System
smss.exe	0	System
csrss.exe	0	System
csrss.exe	1	System
wininit.exe	0	System
services.exe	0	System
svchost.exe	0	System
svchost.exe	0	System
nssm.exe	0	System
conhost.exe	0	System
cmd.exe	0	System
docker.exe	0	System
svchost.exe	0	System
MsMpEng.exe	0	System
vmms.exe	0	System
svchost.exe	0	System
svchost.exe	0	System
vmcompute.exe	0	System
CcmExec.exe	0	System
msdtc.exe	0	System
svchost.exe	1	Medium
svchost.exe	2	Medium
WmiApSrv.exe	0	System
lsass.exe	0	System
winlogon.exe	1	System
csrss.exe	2	System
winlogon.exe	2	System
dwm.exe	2	System
explorer.exe	1	Medium
cmd.exe	1	Medium
conhost.exe	1	Medium
docker.exe	1	Medium
cmd.exe	1	Medium
conhost.exe	1	Medium
ServerManager.exe	1	High

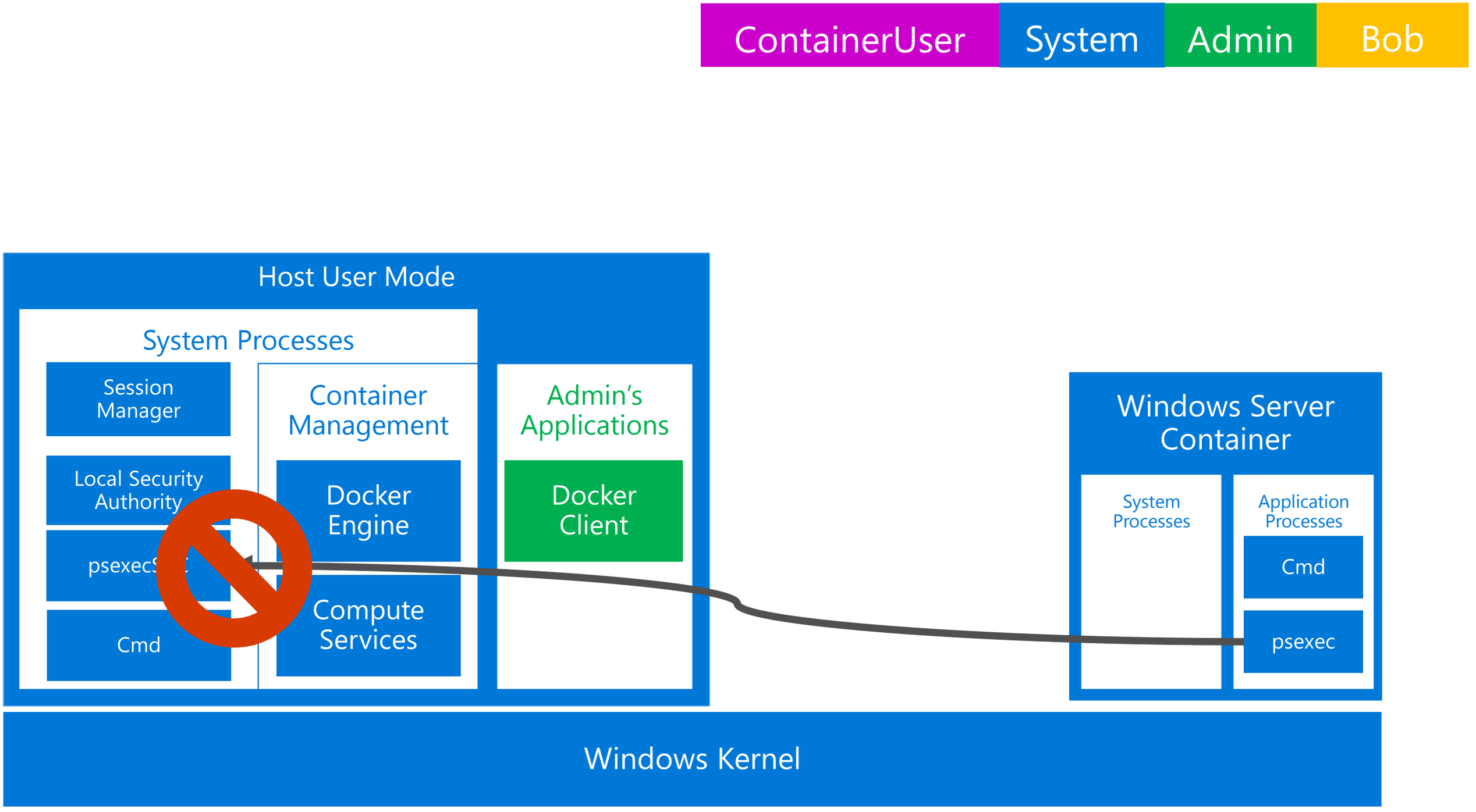
Process	Session	Integrity
csrss.exe	3	System
wininit.exe	3	System
services.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
svchost.exe	3	System
CExecSvc.exe	3	System
cmd.exe	3	System
msdtc.exe	3	System
sppsvc.exe	3	System
lsass.exe	3	System

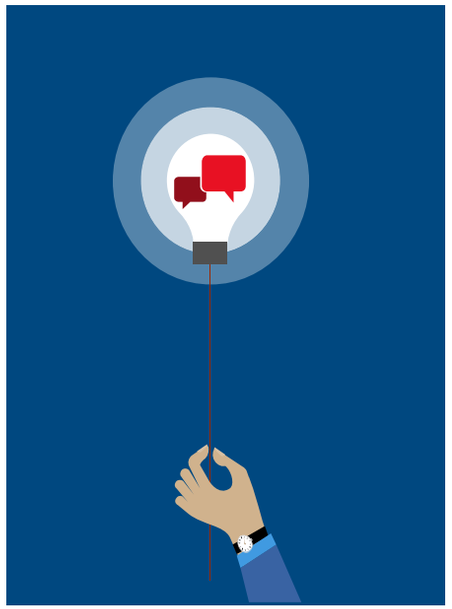
```
C:\Windows\System32\>psexec \\localhost\ cmd
```

System | Bob




Host
ystem
Bob
ows Server
ntainer
Sessi
Mana
Local Se
Autho
psexec
Cm
Windows Kernel
Command Prompt - docker run -it windowsservercore cmd
C:\Windows\system32>C:\psexec.exe \\localhost\ /accepteula cmd
PsExec v2.2 - Execute processes remotely
Copyright (C) 2001-2016 Mark Russinovich
Sysinternals - www.sysinternals.com
Microsoft Windows [Version 10.0.10586]
(c) 2015 Microsoft Corporation. All rights reserved.
C:\Windows\system32>whoami
nt authority\system
C:\Windows\system32>dir C:\users
Volume in drive C has no label.
Volume Serial Number is 755F-FCA0
Directory of C:\users
10/20/2016 01:59 PM <DIR> .
10/20/2016 01:59 PM <DIR> ..
10/20/2016 11:59 AM <DIR> Public
10/20/2016 02:00 PM <DIR> skarade
10/20/2016 11:59 AM <DIR> TDPUser
0 File(s) 0 bytes
5 Dir(s) 23,430,877,184 bytes free
C:\Windows\system32>
Command Prompt
C:\Users\TDPUser>whoami
skarade-th2\tdpuser
C:\Users\TDPUser>
Process
System Idle Process
System
Interrupts
smss.exe
smss.exe
csrss.exe
csrss.exe
wininit.exe
services.exe
svchost.exe
nssm.exe
svchost.exe
MsMpEng.exe
vmms.exe
svchost.exe
svchost.exe
vmcompute.exe
CcmExec.exe
msdtc.exe
policyHost.exe
svchost.exe
svchost.exe
PSEXESVC.exe
cmd.exe
conhost.exe
lsass.exe
winlogon.exe
csrss.exe
winlogon.exe
explorer.exe
cmd.exe
conhost.exe
docker.exe
cmd.exe
conhost.exe
ServerManager.exe
2 System
1 Medium
1 Medium
1 Medium
1 Medium
1 Medium
1 Medium
1 High
Session
Integrity
3 System
psexec.exe
msdtc.exe
sppsvc.exe
lsass.exe

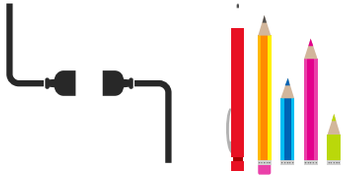

ContainerUser
System
Admin
Bob
Host User Mode
System Processes
Session Manager
Local Security Authority
psexecs
Cmd
Container Management
Docker Engine
Compute Services
Admin's Applications
Docker Client
Windows Server Container
System Processes
Application Processes
Cmd
psexec
Windows Kernel



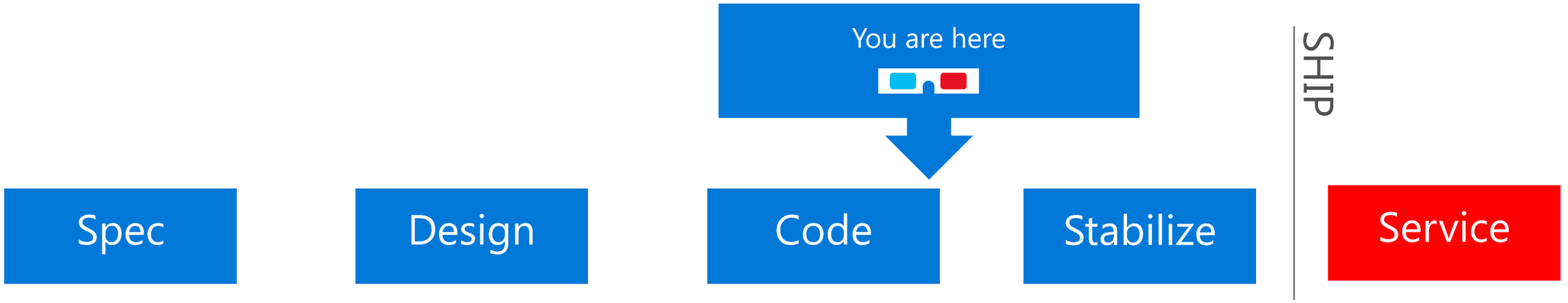
1. Require Admin privileges to communicate to dockerd.exe by default.

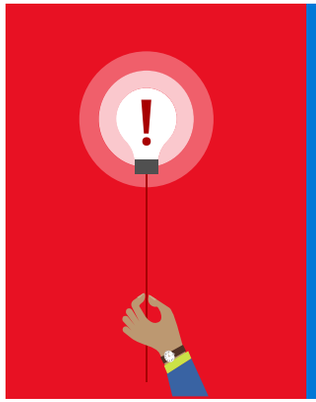
2. Find and block gaps in isolation.

3. Run processes inside the Container as "ContainerUser", not SYSTEM.



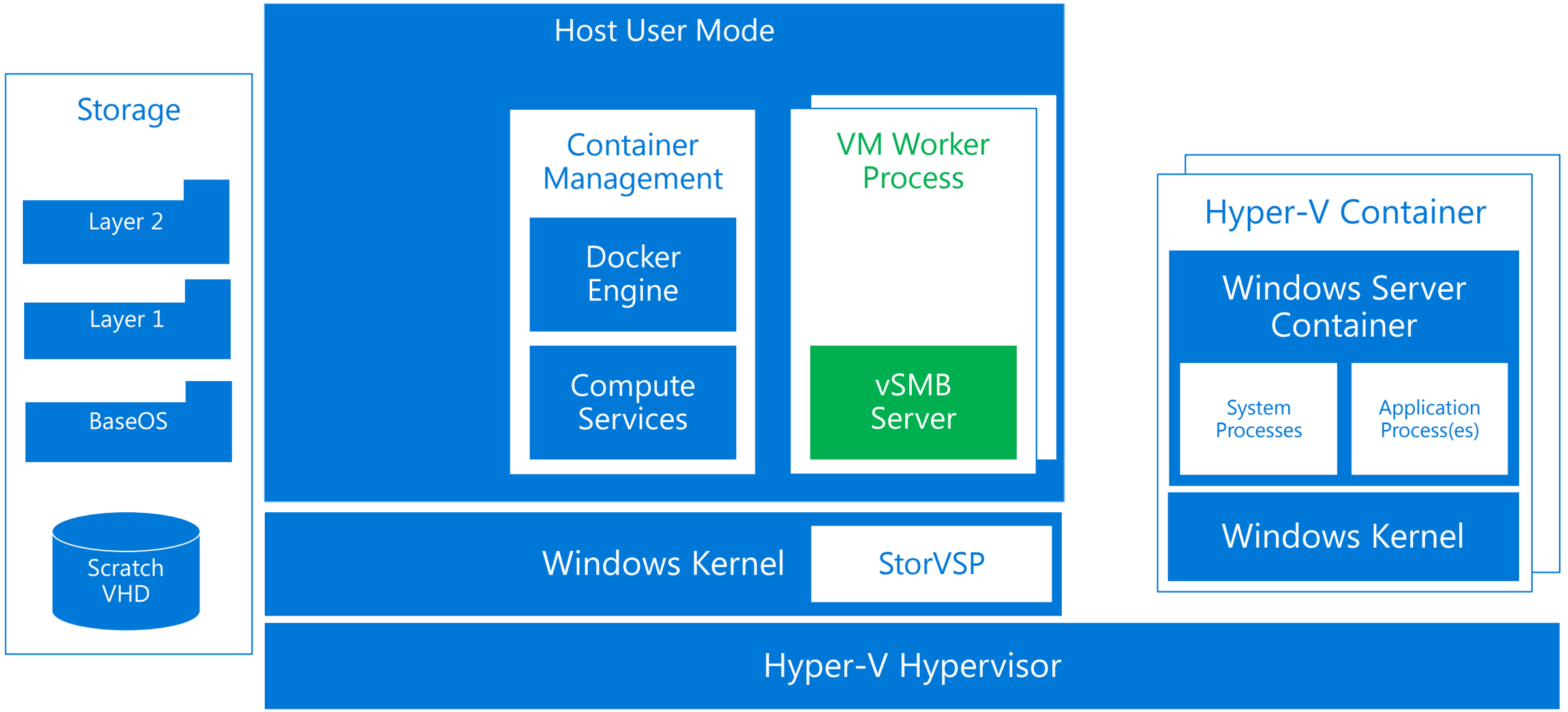
Story 3: Storage Architecture





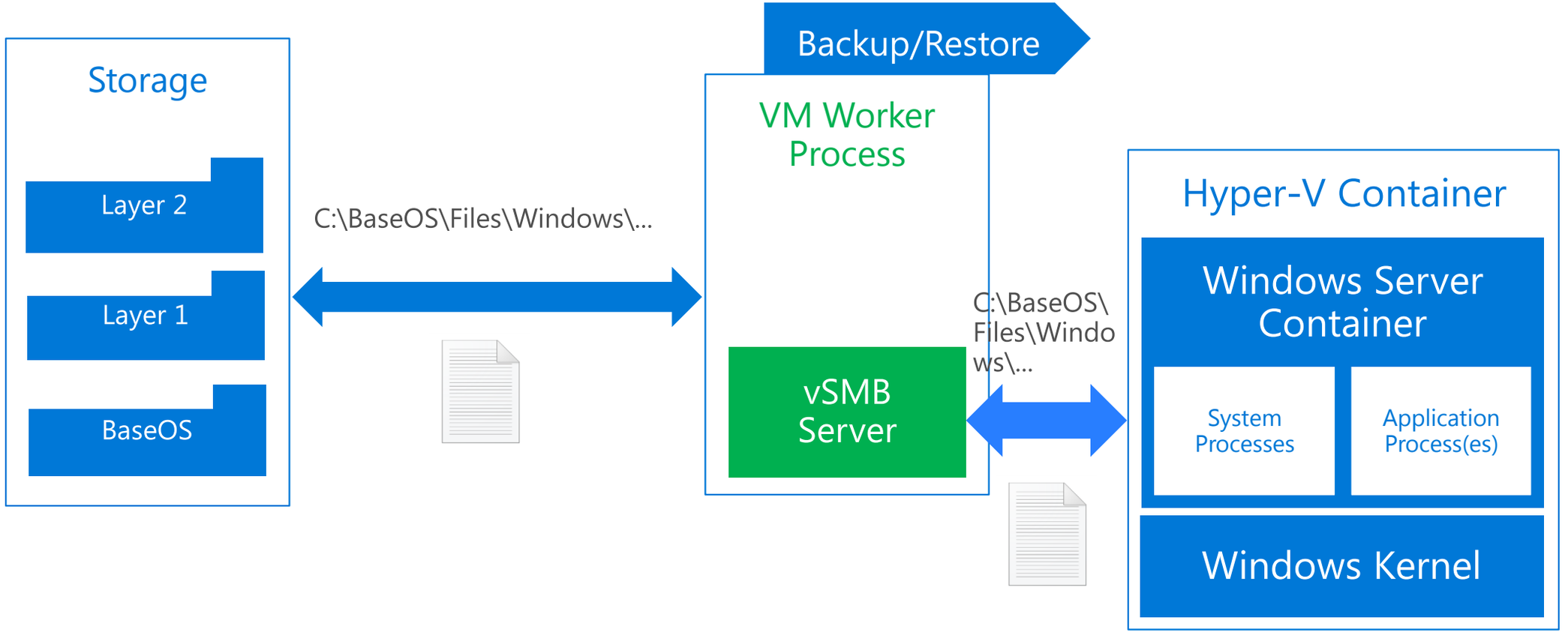
File system access privileges granted to high risk VMWP process that handles untrusted input.

Storage architecture



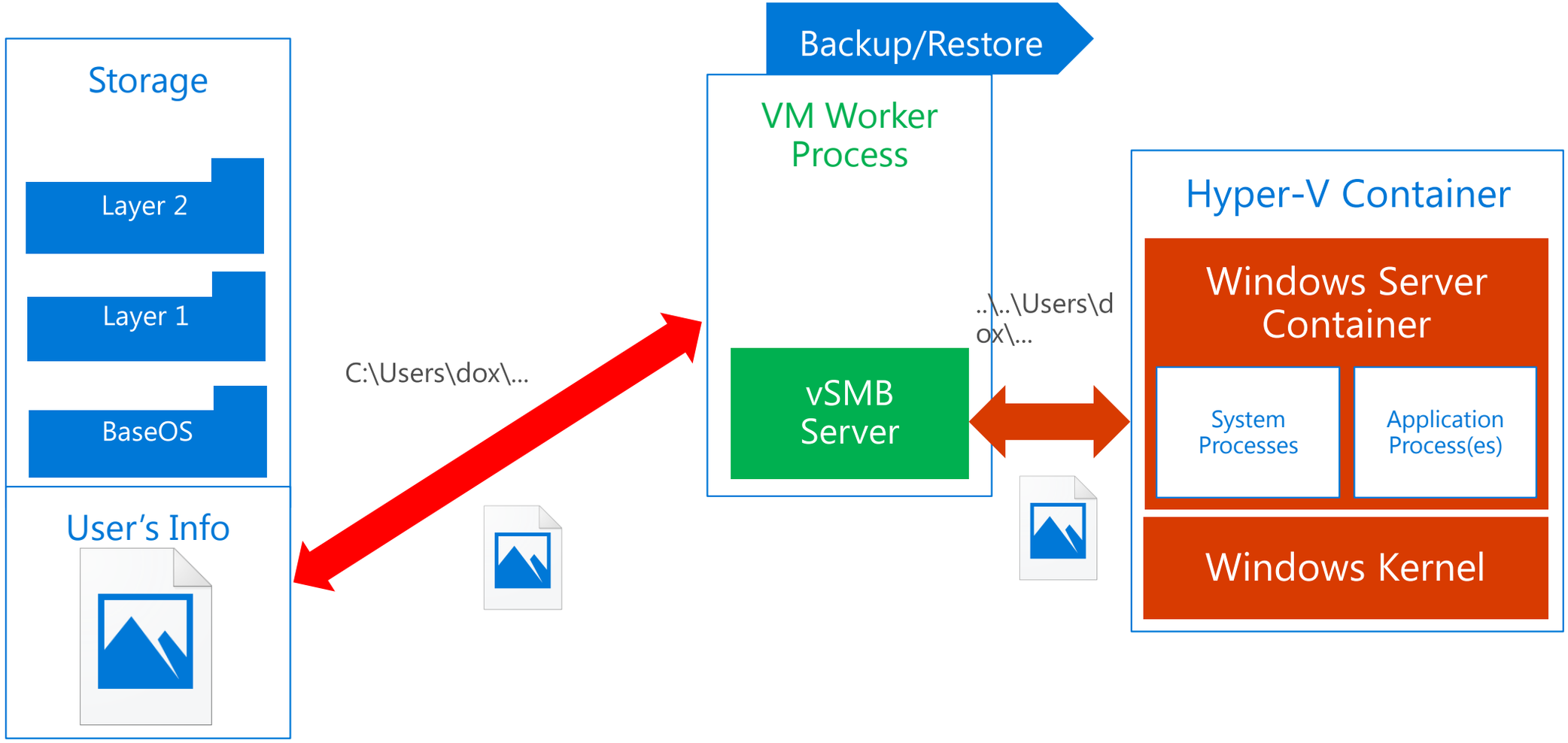
Before

System | VirtualMachine



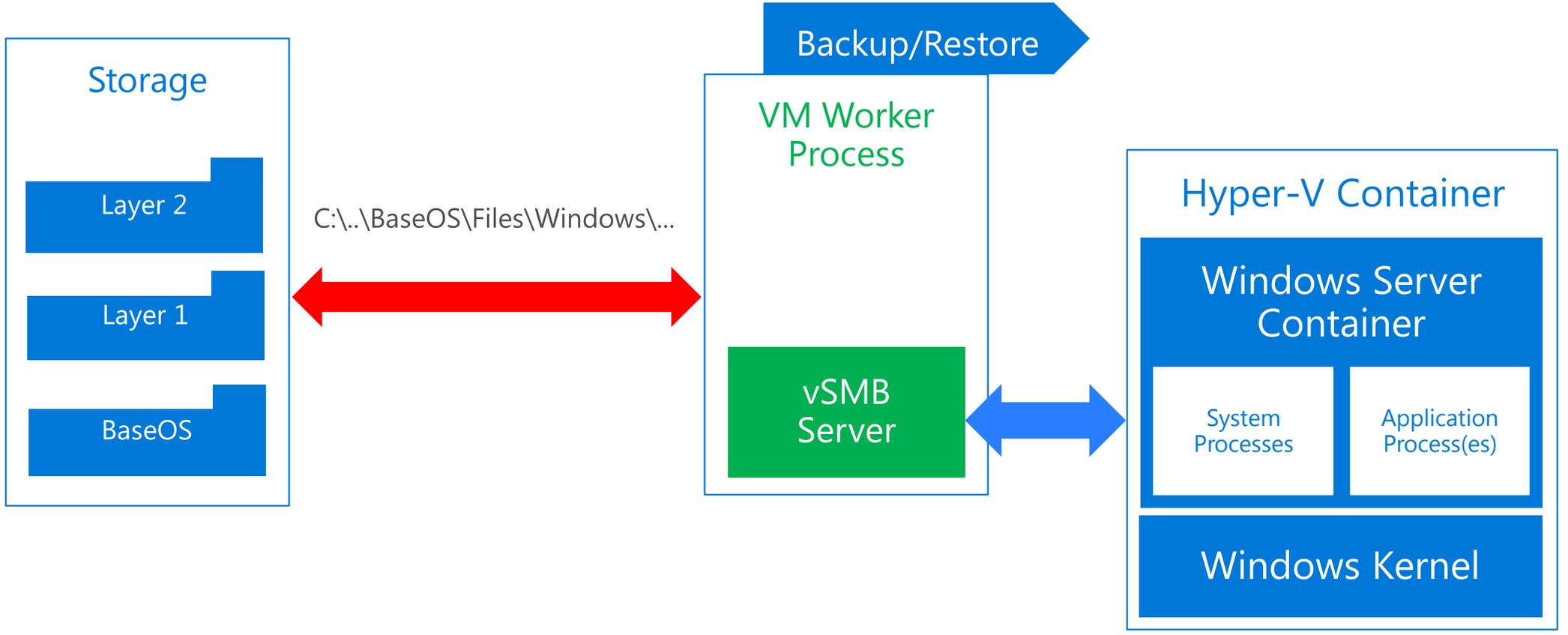
Before

System | VirtualMachine

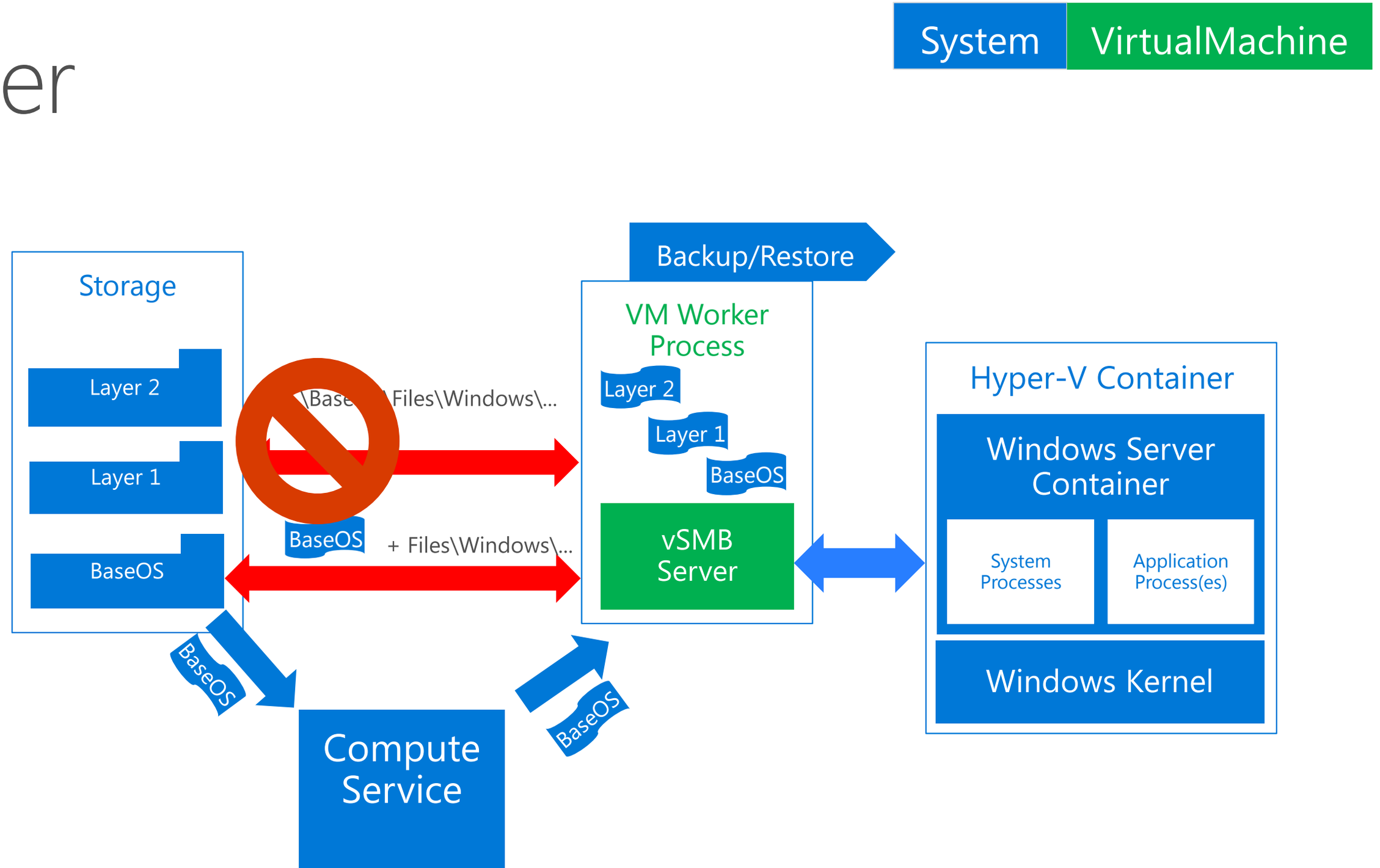


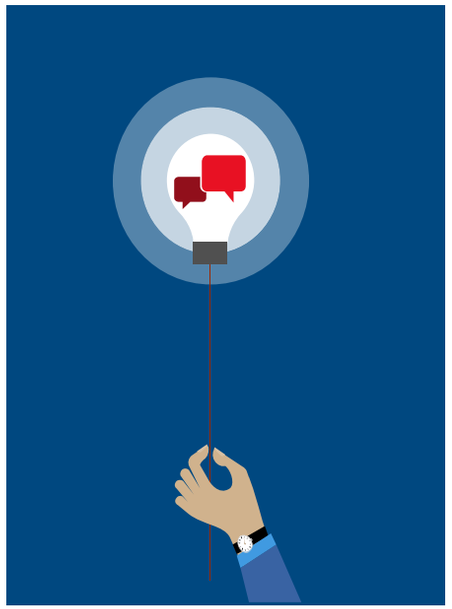
Before

System VirtualMachine



After


System
VirtualMachine
Backup/Restore
Storage
Layer 2
Layer 1
BaseOS
\Base Files\Windows\...
BaseOS
+ Files\Windows\...
VM Worker Process
Layer 2
Layer 1
BaseOS
vSMB Server
Hyper-V Container
Windows Server Container
System Processes
Application Process(es)
Windows Kernel
BaseOS
BaseOS
Compute Service



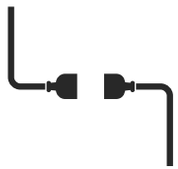
1. Open a handle to the target folder in a trusted process.

2. Pass the folder handle to VMWP to limited areas and components.

3. Read files using folder handle while unable to read anywhere else.

Start early

Engage teams



Provide feedback

Stay informed

Mitigation Bypass and Bounty for Defense

A security mitigation improves on the security of our products

Submit a novel mitigation bypass against our latest Windows platform, and/or a defense idea that would block an exploitation technique that currently bypasses the latest platform mitigations

- Stack corruption (/GS, SEHOP, and SafeSEH)
- Heap corruption (metadata integrity checks)
- Code execution (DEP, CFG and ASLR)

Total payout range is: Up to $200,000
(Mit. Bypass + Bounty for Defense)

For additional information about this program: https://technet.microsoft.com/en-us/dn425049

Hyper-V

Hyper-V escapes that will receive a bounty

- Guest-to-Host
- Guest-to-Guest
- Guest-to-Host DoS (non-distributed, from a single guest)

Total payout range is: Up to $100,000 USD

For additional information about this program: https://technet.microsoft.com/en-us/dn425049

Bounties Paid To Date

- Mitigation Bypass, Bounty for Defense and BlueHat Prize
  > $600,000 USD
- Online Services Bug Bounty
  > $400,000 USD
- Software Bounties
  > $200,000 USD

CVD: Coordinated Vulnerability Disclosure

- We request that you keep customers secure by maintaining the confidentiality of the vulnerability report to MSRC
- If you wish to discuss the vulnerability publically or blog about it, please wait till it has been fixed and patches have been released to customers
- Preferably, blog or present the vulnerability 30 days after it has been patched. This gives customers enough time to take the patch
- Never publish any exploit code (please ☺)
- We are happy to provide technically review to any talks, white papers or blogs you are publishing

For additional information about this program: https://technet.microsoft.com/en-us/security/dn467923.aspx

Take Action

1. Visit https://aka.ms/BugBounty for a current list of active bounties
2. Identify the bounty you want to go after and start hacking away at it
3. Report your findings to secure@microsoft.com
   - Describe the bug and how you exploit it
   - Provide a Proof of Concept (PoC)
   - For complicated bugs (software) provide a white paper or detailed write up
     - If it's a high quality report, you get larger bounties
     - If it has greater impact to Microsoft, you get larger bounties
4. Give us your name and a good email to reach you at
5. Encrypt with our public key (if it's a PoC or working exploit)
6. For eligible bounty cases, GET PAID!

Shout-outs!

Hyper-V: Kevin Broas, Bruce Sherwin, Mike Ebersol, Matt Kurjanowicz, Lars Reuther, John Starks, Martijn de Kort, Arseney Romanenko, John Howard, Stefan Wernli, Taylor Brown,

Kernel: Erick Smith, John Richardson

WDG Client Pentest: Jonathan Norman, Logan Gabriel, Adam Zabrocki, Mary Lee

Thank you!

Questions?

Resources

https://docs.com/taylorbrown/1326/windows-containers-ignite

https://goto.docker.com/Definitive-Guide-to-Docker-Whitepaper-LP.html