Some things in Nanos are set in stone and others are not. In general security and performance are top of mind and we abide by KISS principles.

Quick FYI: This site is mainly for Nanos specific information. If you are an end-user and you just want more "getting started" docs please check out the DOCS on OPS.CITY which are substantial.

This site is a WIP (work in progress).

```
Filesystem
Networking
Performance
Security
Architecture
Infrastructure
Syscalls
Features
Tools
Manifest
Data Structures
```

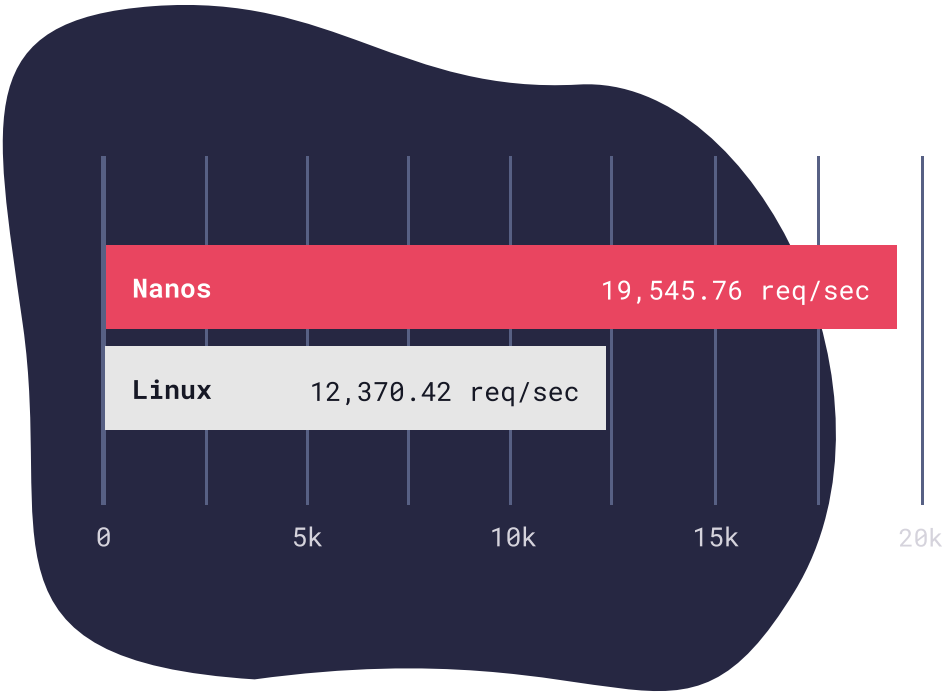
The filesystem currently used by Nanos is TFS. Nanos isn't opposed to other file systems but hasn't identified a large need yet either. As with most of these sections if your team requires different filesystem support please reach out to the NanoVMs team for a

support subscription.

For more info on the TFS filesystem.

Nanos supports both IPV4 and IPV6. For more information on configuring things like VPCs, firewalls and the like please consult the OPS networking config pages for your specific cloud.

Not a lot of benchmarking and tuning has been done yet, however, there is plenty of potential. Currently, our naive tests can push 2X the amount of requests/second for Go webservers. This website is hosted on a Go webserver running a recent 0.1.27 version of Nanos. We've also seen up to 3X improvements on AWS.



Nanos has an opionated view of security. Users and their associated permissions are not supported. Nanos is also a single process (but multi-threaded) system. This means there is no support for SSH, shells or any other interactive multiple command/program running.

While this prevents quite a few security issues extra precaution should be taken for things such as RFI style attacks. For instance you wouldn't want to leak your SSL private key or database credentials.

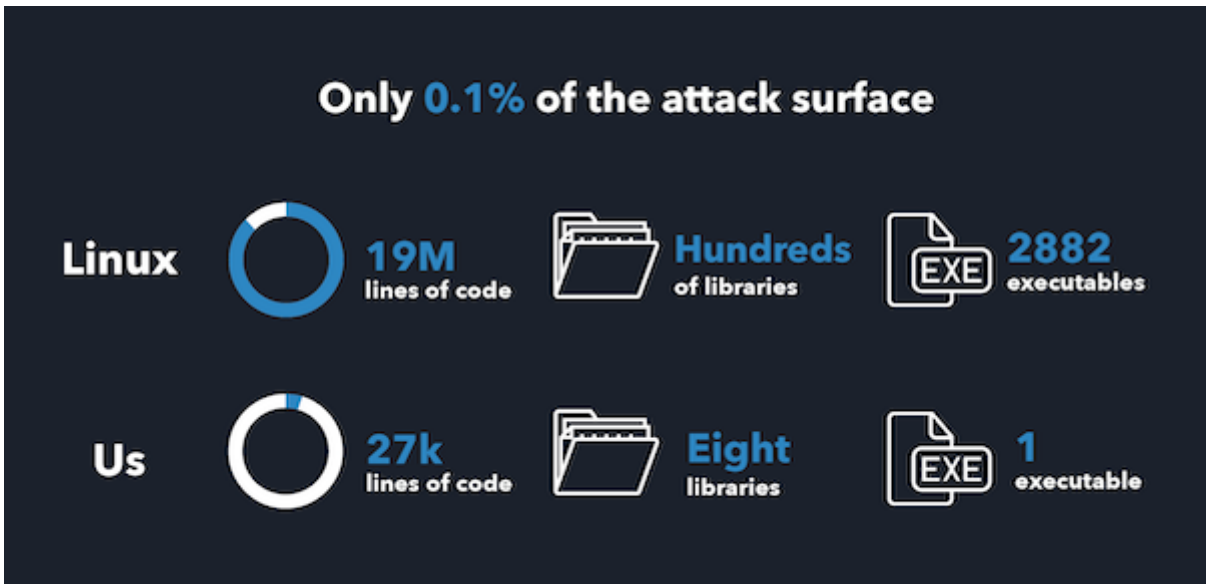
Similarily, just cause you can't create a new process doesn't mean an attacker couldn't inject their process.

Nanos employs various forms of security measures found in other general purpose operating systems including ASLR and respects page protections that compilers produce.

Nanos, unlike other general purpose operating systems, only provision what is necessary on the filesystem to run an application so most filesystems will have a few to maybe 10 libraries and many applications might have filesystems with only a handful of files on them.

Nanos's kernel lives on a different partition and is separated from the user-viewable partition. Nanos goes further with the idea of exec protection with an optional exec_protection flag available in the manifest. When this is enabled the application cannot modify the executable files and cannot create new executable files. For further information check out this PR.

For more info: more info

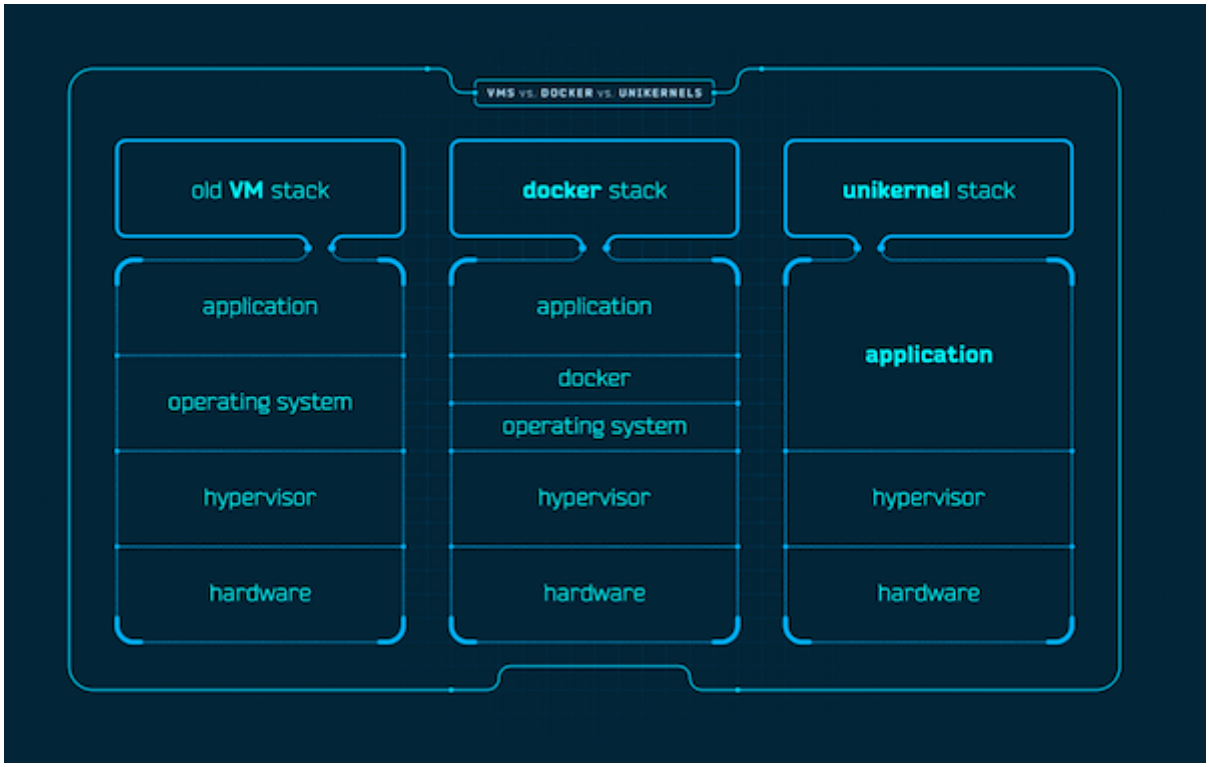


Nanos reduces its attack surface through a variety of thrusts. Compared to a normal Ubuntu or Debian instance has multiple orders

of magnitude less lines of code, libraries only that are needed by an application and thousands of less executables - in fact it only can run one.

Currently Nanos only targets X86-64 and has limited ARM64 support, specifically for the rpi4.

RISC-V and the POWER family of architectures have been asked for but so far there is no roadmap for it. If you are interested in getting that sooner reach out to the NanoVMs team.



Nanos is always deployed as a guest VM directly on top of a hypervisor. Unlike Linux that runs many different applications on top of it Nanos molds the system and application into one discrete unit. Unlike Containers that duplicate storage and networking layers with an orchestrator in between Linux and the application Nanos relies on the native storage and networking layers present in the hypervisor of choice.

Nanos can currently deploy to the following public cloud providers:

→ Google Cloud
→ Amazon Web Services
→ Digital Ocean
→ Vultr
→ Microsoft Azure
→ Oracle Cloud
→ UpCloud

Nanos can also deploy to the following hypervisors:

→ KVM
→ Xen
→ ESX
FireCracker
VirtualBox
→ Hyper-V

Nanos can even run on K8S.

Syscalls

Supported:

socket
bind
listen
accept
accept4
connect
sendto
sendmsg
sendmmsg
recvfrom
recvmsg
setsockopt
getsockname
getpeername

```
      getsockopt
      shutdown
      futex
      clone
      arch_prctl
      set_tid_address
      gettid
      timerfd_create
      timerfd_gettime
      timerfd_settime
      timer_create
      timer_settime
      timer_gettime
      timer_getoverrun
      timer_delete
      getitimer
      setitimer
      alarm
      mincore
      mmap
      mremap
      msync
      munmap
      mprotect
      epoll_create
      epoll_create1
      epoll_ctl
      poll
      ppoll
      select
      pselect6
      epoll_wait
      epoll_pwait
      read
      pread64
      write
      pwrite64
      open
      openat
      dup
```

```
    dup2
    dup3
    fstat
    fallocate
    fadvise64
    sendfile
    stat
    lstat
    readv
    writev
    truncate
    ftruncate
    fdatasync
    fsync
    sync
    syncfs
    io_setup
    io_submit
    io_getevents
    io_destroy
    access
    lseek
    fcntl
    ioctl
    getcwd
    symlink
    symlinkat
    readlink
    readlinkat
    unlink
    unlinkat
    rmdir
    rename
    renameat
    renameat2
    close
    sched_yield
    brk
    uname
    getrlimit
```

```
        setrlimit
        prlimit64
        getrusage
        getpid
        exit_group
        exit
        getdents
        getdents64
        mkdir
        mkdirat
        getrandom
        pipe
        pipe2
        socketpair
        eventfd
        eventfd2
        creat
        chdir
        fchdir
        utime
        utimes
        newfstatat
        sched_getaffinity
        sched_setaffinity
        capget
        prctl
        sysinfo
        umask
        statfs
        fstatfs
        io_uring_setup
        io_uring_enter
        io_uring_register
        kill
        pause
        rt_sigaction
        rt_sigpending
        rt_sigprocmask
        rt_sigqueueinfo
        rt_tgsigqueueinfo
```

```
rt_sigreturn
rt_sigsuspend
rt_sigtimedwait
sigaltstack
signalfd
signalfd4
tgkill
tkill
clock_gettime
clock_nanosleep
gettimeofday
nanosleep
time
times
```

unsupported:

```
shmget
shmat
shmctl
fork
vfork
execve
wait4, syscall_ignore);
semget
semop
semctl
shmdt
msgget
msgsnd
msgrcv
msgctl
flock, syscall_ignore);
link
chmod, syscall_ignore);
fchmod, syscall_ignore);
fchown, syscall_ignore);
lchown, syscall_ignore);
ptrace
syslog
getgid, syscall_ignore);
getegid, syscall_ignore);
```

```
setpgid
getppid
getpgrp
setsid
setreuid
setregid
getgroups
setresuid
getresuid
setresgid
getresgid
getpgid
setfsuid
setfsgid
getsid
mknod
uselib
personality
ustat
sysfs
getpriority
setpriority
sched_setparam
sched_getparam
sched_setscheduler
sched_getscheduler
sched_get_priority_max
sched_get_priority_min
sched_rr_get_interval
mlock, syscall_ignore);
munlock, syscall_ignore);
mlockall, syscall_ignore);
munlockall, syscall_ignore);
vhangup
modify_ldt
pivot_root
_sysctl
adjtimex
chroot
acct
```

```
settimeofday
mount
umount2
swapon
swapoff
reboot
sethostname
setdomainname
iopl
ioperm
create_module
init_module
delete_module
get_kernel_syms
query_module
quotactl
nfsservctl
getpmsg
putpmsg
afs_syscall
tuxcall
security
readahead
setxattr
lsetxattr
fsetxattr
getxattr
lgetxattr
fgetxattr
listxattr
llistxattr
flistxattr
removexattr
lremovexattr
fremovexattr
set_thread_area
io_cancel
get_thread_area
lookup_dcookie
epoll_ctl_old
```

```
        epoll_wait_old
        remap_file_pages
        restart_syscall
        semtimedop
        clock_settime
        vserver
        mbind
        set_mempolicy
        get_mempolicy
        mq_open
        mq_unlink
        mq_timedsend
        mq_timedreceive
        mq_notify
        mq_getsetattr
        kexec_load
        waitid
        add_key
        request_key
        keyctl
        ioprio_set
        ioprio_get
        inotify_init
        inotify_add_watch
        inotify_rm_watch
        migrate_pages
        mknodat
        fchownat, syscall_ignore);
        futimesat
        linkat
        fchmodat, syscall_ignore);
        faccessat
        unshare
        set_robust_list
        get_robust_list
        splice
        tee
        sync_file_range
        vmsplice
        move_pages
```

```
utimensat
inotify_init1
preadv
pwritev
perf_event_open
recvmmsg
fanotify_init
fanotify_mark
name_to_handle_at
open_by_handle_at
clock_adjtime
setns
getcpu
process_vm_readv
process_vm_writev
kcmp
finit_module
sched_setattr
sched_getattr
seccomp
memfd_create
kexec_file_load
bpf
execveat
userfaultfd
membarrier
mlock2, syscall_ignore);
copy_file_range
preadv2
pwritev2
pkey_mprotect
pkey_alloc
pkey_free
```

Features

→ -d strace
→ ftrace
→ http server dump

Tools

Several tools are packaged inside Nanos: mkfs

```
➜  ~ ~/.ops/0.1.27/mkfs -help
/Users/eyberg/.ops/0.1.27/mkfs: illegal option -- h
Usage:
mkfs [options] image-file < manifest-file
mkfs [options] -e image-file
Options:
-b boot-image   - specify boot image to prepend
-k kern-image   - specify kernel image
-r target-root  - specify target root
-s image-size   - specify minimum image file size; can be expressed in
bytes, KB (with k or K suffix), MB (with m or M suffix), and GB (with g
or G suffix)
-e              - create empty filesystem
```

dump

```
➜  ~ ~/.ops/0.1.27/dump
Usage: dump [OPTION]...
Options:
  -d        Copy filesystem contents from  into
  -t                   Display filesystem from  as a tree
```

There are also development tools available such as plugins for various editors: OPS for Visual Studio
IntelliJ

Manifest

The nanos manifest is an extremely powerful tool as it comes with many different flags and is the synthesis of a filesystem merged with various settings. Most users will never craft their own manifests by hand, opting to use OPS to craft it automatically.

→ futex_trace
→ debugsyscalls

→ fault
→ exec_protect

Data Structures

Nanos uses a variety of internal data structures. This is only a partial list.

- Bitmap
- ID Heap
- FreeList
- Backed Heap
- Linear Backed Heap
- Paged Back Heap
- Priority Queue
- RangeMap
- Red/Black Tree
- Scatter/Gather List
- Table
- Tuple