Existing Options

- Serverless model attractive:
  - Automatic scaling
  - Pay for use pricing
  - Reduced cost of operations
  - Improved hardware utilization
- Multitenancy presents significant challenges in isolating workloads
  - Workloads cannot access / infer data belonging to another
  - Noisy neighbors cannot cause others to run more slowly
- Hypervisor-based virtualization
  - High overhead
- Linux containers
  - Built into Linux but limits to only using Linux and also restricts system calls
  - Leads to a tradeoff between security and code compatability
- QEMU: type 2 hypervisor that emulates hardware but to emulate CPUs it has to emulate every instruction
  - Also: VirtualBox / VMWARE
- KVM: type 1 hypervisor that is a Linux kernel module that uses modern CPU features to directly run vCPU instructions on the CPU
- chroot:
  - Command used to make a subprocess treat a different directory as its root directory

Isolation Options

- Previously, each customer would get a single VM and all of their functions would run in that
  - Made it hard to efficiently pack workloads
- Desired features:
  - Isolation
  - Overhead and density: run thousands of functions with minimal waste
  - Performance: similar to native performance and is consistent
  - Compatability: allow arbitrary Linux binaries / libraries without code changes
  - Fast switching: start / clean up functions quickly
  - Soft allocation: can over commit CPU and have functions consume only resources it needs, not that it is entitlted to
- Linux options:
  - Containers - all workloads share kernel and kernel mechanisms isolate them
    - Does this primarily by limiting the syscalls a workload can use, which hurts compatability
  - Virtualization - all workloads run in a VM under a hypervisor
    - Leads to higher overhead / lower density because the VMM and kernel associated with each VM leads to overhead
    - Could use smaller kernels to mitigate this but that hurts compatability
  - Language VM isolation - use something like JVM to isolate but this is difficult to run arbitrary code on

Firecracker

- Builds on top of KVM but replaces QEMU
- Built specifically for serverless and container applications
  - Does not offer a BIOS, cannot boot arbitrary kernels (i.e. Windows)
  - Doesn't offer VM orchestration / packaging (instead handled by Docker / Kubernetes / others)
- A single Firecracker process runs per MicroVM, which is a minimal virtual machine offered by KVM
- Builds substantially on top of Linux functionality that limits the compatability but offers comfort and insurance that this has good security

- Each process runs in a "jailed" environment with namespaces, limited files / permissions, etc. to prevent access to other things
- Uses simplified block filesystem
- Memory safe via Rust
- Exploitable by:
  - Bugs in KVM
  - Bugs in Firecracker hypervisor

gVisor

- An application kernel that implements much of Linux syscall interface to provide isolation between applications and operating system
- Two previous methods:
  - Machine level virtualization (VMs) which emulate hardware and a new kernel to provide strong isolation
    - Large resource footprint
  - Rule-based execution: limit available system calls
    - Hard to apply universally to arbitrary applications
- gVisor acts between the application and host kernel
  - Basically acts as a guest kernel without the need for virtualized hardware
- May provide poor performance for system call heavy workloads
- Two main components:
  - Sentry
    - Intercepts all system calls and then does the necessary work to service it
    - Does not directly pass through system calls
    - Cannot open files directly that are not pipes / internal files
  - Gopher has access to file system resources and communicates with sentry over a secure channel
- Each process interacts with a separate Sentry proess so even if one is compromised, it doesn't give you access to the entire kernel

Comparisons

- Main key: reduce system calls that hit the host kernel
- All use seccomp, which is used to limit the system calls a process can make