Docker Development Trends for Future

The future of Docker

The need to take advantage of modern kernel features in Linux 5.0 and beyond, as well as dealing with different types of new workloads, including stateful workloads, which require a degree of persistence that is not present in stateless workloads. Edge workloads for deployments at the edge of the network, rather than just within a core cloud, are another emerging use case. Internet of Things (IoT) and embedded workloads in small footprint devices and industrial settings are also an important use case for Docker in 2019.

One of the Linux kernel features that Docker will take full advantage of in the future is eBPF, which will someday be usable to write seccomp filters. Crosby explained that seccomp and BPF allow for flexible system call interception within the kernel, which opens the door for new control and security opportunities for containers.

Control groups (cgroups) v2 is another Linux feature that Docker will soon benefit from. Cgroups v2 has been in the kernel since the Linux 4.5 release, though it wasn't immediately adopted as a supported technology by Docker. The project isn't alone in not immediately supporting cgroups v2, Red Hat's Fedora community Linux distribution also has not integrated cgroups v2, though it plans to for the Fedora 31 release that is currently scheduled for November. Crosby said that cgroups v2 will give Docker better resource isolation and management capabilities.

Making containers more stateful

One of the areas that Crosby is most interested in improving is the stateful capabilities of Docker, which in his view are currently relatively limited. Better stateful capabilities include backup, restore, clone, and migrate capabilities for individual containers. Crosby explained that stateful management today in Docker typically relies on storage volumes and not the actual

Rethinking container image delivery

Container images today are mostly delivered via container registries, like Docker Hub for public access, or an internal registry deployment within an organization. Crosby explained that Docker images are identified with a name, which is basically a pointer to content in a given container registry. Every container image comes down to a digest, which is a content address hash for the JSON files and layers contained in the image. Rather than relying on a centralized registry to distribute images, what Crosby and Docker are now thinking about is an approach whereby container images can also be accessed and shared via some form of peer-to-peer (P2P) transfer approach across nodes.