QNET

Internet of Everything Corp

What is QNet:

QNet is a decentralized, autonomous, portable, secure, virtual infrastructure for managing clustered workloads over depos (decentralized pods) and services that facilitates both declarative configuration and automation.

Historically at the beginning of the Internet era, organizations ran applications on physical servers. There was no way to define resource boundaries for services in a physical server, and this caused resource allocation issues. For example, if multiple services ran on a physical server, there could be instances where one service would take up most of the resources, and as a result, the other services would underperform. A solution for this would be to run each service on a different physical server. But this did not scale, as resources were underutilized, and it was expensive for organizations to maintain many physical servers.

So then we all went in for virtualization. It allowed us to run multiple Virtual Machines (VMs) on a single physical server's CPU. Virtualization allows services to be isolated between VMs and provides a level of security as the information of one service cannot be freely accessed by another service. Virtualization allows better utilization of resources in a physical server and allows better scalability because a service can be added or updated easily, reduces hardware costs, and much more. With virtualization, you can present a set of physical resources as a cluster of disposable virtual machines. Each VM is a full machine running all the components, including its own operating system, on top of the virtualized hardware.

Then came the container deployment era, containers are similar to VMs, but they have relaxed isolation properties to share the Operating System (OS) among the applications. Therefore, containers are considered lightweight. Similar to a VM, a container has its own filesystem, the share of CPU, memory, process space, and more. As they are decoupled from the underlying

TechConnect Briefs 2021, TechConnect.org, ISBN 978-0-578-99550-2

infrastructure, they are portable across clouds and OS distributions.

When we looked at this development we noticed one limiting factor for the future, and that is that you always need to move the data to a centralized point, this works great for web services that are inherently centralized, but for future massive IoT or mobile App solutions, it does not make that much sense to move data from a load of devices to a central point and then move the data back out to the devices again.

So we based QNet on a decentralized model based on scalable device clustering, where it is easy to add in new devices as nodes and made it possible for any device to contribute computing resources over an intelligent mesh network so that computing can happen where it is needed and close to where it will be used. We then developed quantum-safe tunnels using polymorphic encryption keys and used a blockchain with consensus to verify the data moved between the nodes over the tunnels, thus creating trusted data walled gardens. The orchestration of computing and storage is done via service manifests that describe services rules, policies, and logic, the underlying orchestration mechanics is managed by an autonomous knowledge-based AI using network consensus over the blockchain as a deciding mechanism. The cluster topography is dynamically updated by the orchestration to fit the current workload. QNet Service depos are generated and deployed in a similar way to container images; the depos are MPI cluster enabled from start.

Extra benefits:

* Agile service creation and deployment: only addition to a normal container deployment is the Service Manifest generation.
* Continuous development, integration, and deployment: Provides for reliable and frequent depo build and deployment with quick and efficient rollbacks (due to depos immutability).
* Dev and Ops separation of concerns: create service depo's at build/release time rather than deployment time, thereby decoupling services from infrastructure.
* Observability not only cluster information and metrics, but also application health and other signals.
* Environmental consistency across development, testing, and production: Runs the same on a laptop as it does in the wild.
* Cloud and OS distribution portability: Runs on Linux, BSD, on-premises, on public clouds, and anywhere else.
* Service-centric management: Raises the level of abstraction from running an OS on virtual hardware to running a service on a Decentralized network.
* Loosely coupled, distributed, elastic, liberated micro-services: Applications are broken into smaller, independent pieces and can be deployed and managed dynamically – not a monolithic stack running on one big single-purpose machine.
* Resource isolation: predictable service performance.
* Resource utilization: High efficiency and density.

Why you need QNet and what it can do:

Decentralized walled garden systems with depos are a good and secure way to bundle and run your services. In a normal production environment, you need to manage the containers that run your applications and ensure that there is no downtime. For example, if a container goes down, another container needs to start. Wouldn't it be easier if this behavior was handled by the network?
That's how QNet works! QNet provides you with a system and a service framework to run distributed services decentralized and resilient. The QNet orchestrator takes care of scaling and failover for your services, provides deployment patterns, and more. For example, QNet has TestNets where you, for free, can deploy and test your services before going "sharp".

QNet provides you and your users with:

* Defence against denial of service attacks, QNet is fully decentralized DDOS attacks are mitigated, there are no centralized points to takeout.

* Detection of Malware trying to replicate itself to other nodes, by verifying data traffic between nodes over a blockchain, malware can be detected and the infected node identified.
* Bad data and bad player detection using verification and sanity checks on data entering and transported on QNet.
* Service discovery and dynamic load balancing, QNet can expose a service using the Service name from the service manifest or using their QNet service or over their own IP address, even though using the IP directly is now recommended as the security benefits of the data validation is mute. If traffic to service is high, the orchestrator will load balance and distribute the network traffic so that the service is stable.
* Storage, QNet provides access to decentralized storage, be it data-lakes, temporary storage for AI crunches, or persistent storage. It also allows you to mount storage systems of your choice, such as local storage, public cloud providers, and more.
* Automated rollouts and rollbacks, You can describe the desired state (rules, policies, and logic) for your deployed services using Service Manifests, and it can change the actual state to the desired state at a controlled rate. For example, you can automate QNet to create new services for your deployment, remove existing depos and adopt all their resources to the new service.
* Automatic scaling, You provide QNet with the size of the starting cluster of nodes that it can or should use to run service tasks. Then QNet will optimize how much CPU and memory (RAM) each task needs. QNet can with benefit be installed onto your own nodes to make the best use of your resources.
* Self-healing, The QNet Orchestrator restarts depos that fail, replaces depos, kills depos that don't respond to the service manifest-defined health check, and doesn't advertise them to users until they are ready to serve.
* Secret and configuration management, QNet lets you store and manage sensitive information, such as passwords, OAuth tokens, and encryption keys. You can deploy and update secrets and application configuration without rebuilding your depos, and without exposing secrets in your service.

What QNet is not:

QNet is not a traditional, all-inclusive PaaS (Platform as a Service) or IaaS (Infrastructure as a Service) system. Instead, QNet is a secure walled garden solution that operates at the service level rather than at the hardware level, it provides some features common to PaaS and IaaS offerings, such as deployment, scaling, load balancing. However, QNet is not monolithic, and these default solutions are optional and pluggable. QNet provides the building blocks for building and deploying service, but preserves user choice and flexibility where it is important.

QNet:
* Does not limit the types of services supported. QNet aims to support an extremely diverse variety of workloads, including stateless, stateful, and data-processing workloads. If a service can run from a container image, it should run great on a depo.
* QNet does not deploy source code and does not build your application. Continuous Integration, Delivery, and Deployment (CI/CD) workflows are determined by organization cultures and preferences as well as technical requirements.
* Does not dictate logging, monitoring, or alerting solutions except for security concerns. It provides integrations as help, and mechanisms to collect and export metrics.
* Does not provide nor mandate a configuration language/system. It provides a declarative Service Manifest system.
* Does not provide nor adopt any comprehensive machine configuration, maintenance, management. It uses a Learning Machine AI so that you do not need to worry about the mechanics.
* QNet is though not a mere orchestration system. In fact, in one way it eliminates the need for classic orchestration. The technical definition of

orchestration is the execution of a defined workflow: first, do A, then B, then C. In contrast, QNet comprises a set of independent, composable control AI-controlled processes that continuously drive the current state towards the in the Service Manifest provided desired state. It shouldn't matter how you get from A to C. Centralized control is also not required. This results in a system that is easier to use and more powerful, robust, resilient, and extensible.

Service Example drawing: