NCN/A3R Native Application Framework (Native-CloudNative/Application as a Resource)

NCN/A3R (hereafter **NA3**) is a **QT**-based application-development framework which prioritizes hybrid solutions combining cloud and desktop/native components. The **NCN** (Native-CloudNative) model refers to desktop client applications that are integrated with Cloud/Native back-ends; by sharing code libraries and data formats across both end-points, **NCN** solutions are more streamlined than native front-ends with generic back-ends, or Cloud/Native back-ends with web-application clients. The **A3R** (Application-as-a-Resource) model promotes self-contained, downloadable applications that can be distributed in source-code fashion and compiled with few (if any) non-**QT** dependencies. The combined **NA3** framework yields a comprehensive application-development toolkit with numerous components to streamline the implementation of **QT** applications (**NA3** can also be used as a template for implementations based on frameworks other than **QT**, such as wxWidgets or Operating-System-specific options).

The Current Status of QT Cloud Integration

There has been considerable demand in the native-application sector for a systematized Cloud Services model designed to interoperate with cross-platform native applications. Cloud/Native components can augment the functionality of native/desktop software by providing remote storage for user data; enabling users to share content for collaborative work; maintaining domain-specific repositories (i.e., spaces of resources whose format is specialized so that only select applications can access them properly); and upgrading or extending applications without re-install. We use the term "Native Cloud/Native" to describe hybrid applications whose server and client endpoints are both internally native – in contrast to conventional Cloud/Native where native servers are paired with (potentially) non-native clients.

Cloud/Native support in existing native-application frameworks is fairly primitive. Since **QT** is by far the most widely-used such framework, the **QT** case is instructive. In 2013 (following an earlier beta phase) the **QT** company introduced

Qt Cloud Services, which provided a convenient, Qt-aware cloud-hosting platform for **QT** accounts (in the company's words:

Qt Cloud Beta has solved an immense need for Qt developers when it comes to backend-as-a-service and believe that there is an even greater need to provide the Qt ecosystem with an all-in-one Qt solution for cloud computing). However — to the consternation of the Qt community — this project was discontinued several years later (retroactively we can identify some design flaws which might have hindered the project). Meanwhile, OpenShift discontinued their free-tier Cloud/Native hosting last year, and another company with Cloud/Native options, Arukas, is folding at the end of this month. This means that **QT** developers have limited options even for hosting hand-rolled **QT** cloud solutions (which can be done by compiling **QT** into a Ubuntu container)

Considering the prominence of both **QT** and Cloud/Native technologies in the contemporary computing landscape, it is disconcerting that no standard framework or hosting service provides a cloud platform which works with **QT** "out-of-the-box." The existence of such a platform would be a boon to software in sectors like scientific computing, bioinformatics, bioimaging, pharmaceuticals, academic publishing, and

other fields (where due to complex GUI and/or data-analytic requirements) the software is predominantly native-compiled and desktop-oriented.

Of course, many desktop applications have some web integration, but the current architecture forces the client-facing and web-facing components of the application to be almost completely separate, which adds to development time and expense. Moreover, current native-application environments do not fully leverage Cloud/Native services; they may well be implemented via more old-fashioned non-cloud servers. The great possibility of Native Cloud/Native is a peer-to-peer client-server relationship, sharing libraries and data formats on both ends; and the infrastructure to bring the benefits of Cloud Computing (e.g. faster development and deployment, and less expensive hosting, as compared to non-cloud web services) to the native-application sector.

Native Cloud/Native in the context of NA3

In light of the limitations just identified, LTS intends to contribute tools or hosting arrangements that would bring some of the capabilities of QT Cloud Services back to the market. The simplest commercial model for such a product is to licence a containerized QT-based HTTP server that can run as a local application during testing and development, before being deployed to a container hosting service. We have implemented a prototype server along these lines that we call NDP CLOUD (for "Native-Driven Platform"). NDP-CLOUD is fully self-contained in a QT context (it bundles portions of the Node.js code base and utilizes the QT network module, so it requires no external HTTP or sockets libraries). One significant benefit of NDP CLOUD is that it is fully transparent: all of the code for parsing and routing HTTP requests can be loaded into IDEs (such as QT creator) and examined by the debugger. Another benefit is that project-specific libraries can be compiled into both NDP CLOUD instances and client front-ends; therefore, clients and servers can share procedures for serializing and deserializing domain-specific data structures. For development and prototyping, NDP CLOUD can be launched as an ordinary (non-virtual) OS; further testing can then be performed running NDP CLOUD as a local Docker container, before eventually deploying the application to a remote Docker hosting environment.

Via NDP CLOUD, NCN applications can be deployed on any Docker cloud service, such as OpenShift. In this guise LTS has no direct involvement with the hosting service (although NDP CLOUD includes some tools to streamline cloud deployment). Ideally, however, LTS would like to secure its own hosting capabilities, perhaps by using an LTS-specific container depoyed on OpenShift or a similar platform. LTS would allocate cloud assets to NDP CLOUD licensees (e.g. a limited free-tier hosting plan) for testing and development. A further possibility is to provide free hosting, subject to data-space constraints, to scientific institutions. The dwindling availability of free-tier Cloud/Native options is a hinderance to projects' adoption of Cloud/Native solutions for sharing and disseminating scientific data; this can result in researchers hosting data sets on platforms such as Mendeley or DataVerse, which have limited functionality or customizability compared to Cloud/Native containers. Use-cases for NA3 in the context of scientific data sets are explained in the discussion of A3R below.

Application Development via A3R

The A3R

The A3R