Conformance Testing Cloud Foundry PaaS Certification Requirements

Current work in the Cloud Foundry community on Kubernetes-native implementations, and the tendency for unavoidable divergence between implementations, will require some changes in our conception of how Cloud Foundry platforms are certified by the Foundation. The current “exact software” packaging requirements would have to be broadened to include both KubeCF and CF-for-K8s, but this alone might not actually guarantee consistency of behavior for the end user.

The Kubernetes community has implemented their Kubernetes Software Conformance Certification as a test-based process, which has been very successful. There are currently over 90 certified Kubernetes distributions.

Since the Cloud Foundry Acceptance Tests (CATS) already exist to validate the behavior of the platform during development, we propose to adopt this as the core and starting point for a similar conformance-based test suite for Cloud Foundry Certified Platforms.

Current State of CATs

TODO: Describe structure of CATS and where they might need improvements to make testing easier (e.g. excluding certain destructive tests?).

Other Test Suites

TODO: Are the CATs extensive enough to cover conformance and compatibility? Gather supplemental test suites used by others (e.g. ones used by SUSE or SAP which are broadly applicable).

Conformance Matrix

TODO: Describe core CF functionality vs. implementation-specific functionality. Some distros may choose to implement only core functionality and be able to show which areas are implemented.

APIs

TODO: Discuss whether the entire v2 and/or v3 API must be exposed for certification (expect some input from the community here).

Encouraging Innovation in Cloud Foundry

The “exact software” requirement in the original certification was implemented to encourage collaboration on the extensive Cloud Foundry code base. The intention was to prevent fragmentation of the code base and duplication of effort. While this was somewhat successful, it did discourage more exploratory thinking about the platform design, shut out potentially innovative ideas, and leave some alternative component or full platform implementations stranded with no path to certification.

With recent efforts towards redesigning Cloud Foundry for Kubernetes, we’ve seen a growing difference between new CF-for-K8s components and the “traditional” cf-deployment releases. Updating the certification criteria to simply include CF-for-K8s components would not be sufficient to provide confidence in the community that implementations would remain compatible. There are already differences in behavior between similar v2 and v3 API endpoints and between Diego and Eirini schedulers, so it’s likely more differences would emerge over time unless platform behavior rather than code content becomes the criteria for conformance.

The current certification requirements for software content forced the Quarks+KubeCF teams into an extremely constrained design space, trying to port a VM-based system to Kubernetes as faithfully as possible. This added months of effort and tremendous complexity to the projects. If the acceptance criteria for certification were behavior rather than content, the distribution could have used more “Kubernetes-idiomatic” techniques from the start.

The CF-for-K8s project seems to be free to reshape Cloud Foundry for Kubernetes starting from first-principles. Other projects, within the official Cloud Foundry projects and incubator or even from completely external contributors, should be able to make innovative changes to Cloud Foundry and still have a path for certification and contribution.