# Container Native Development Platform

Marcel Amende, ORACLE Deutschland B.V. & Co. KG

Beim Entwickeln in der Cloud und für die Cloud („Cloud Native Development“) stehen hohe Flexibilität und Interoperabilität im Fokus – bei einfacher Nutzbarkeit und kostengünstigem Betrieb. Derzeitige Top-Themen aus Sicht der Anwender sind die Docker-Containerverwaltung und das sogenannte „Serverless Computing“. Beide Technologien halten Einzug in die Oracle-Cloud-Infrastruktur und sind ein deutliches Zeichen dafür, dass Oracle als Mitglied der „Cloud Native Computing Foundation (CNCF)“ [1] Open-Source-Projekten eine sehr hohe Wertschätzung entgegenbringt.

Neue Technologien krempeln Märkte um, verändern klassische Geschäftsfelder oder eliminieren diese sogar. Unternehmen sind gezwungen, schnell zu handeln. Ein Beispiel: Heute ersetzen Apps wie Uber und Mytaxi die Taxizentralen. Morgen ersetzen autonome Fahrzeuge die Taxifahrer. Man muss sich ständig neu erfinden, digital transformieren, um mit der Veränderungsgeschwindigkeit der Konkurrenz und aufkommenden Start-ups mithalten zu können. Neue Funktionen und neue Geschäftsmodelle müssen entwickelt und in der Praxis erprobt werden. Software ist der Treibstoff dieser Innovation, agile Software-Entwicklung ist gefragt. Ganz wie in der realen Welt bringen Container die neue Ware zum Konsumenten.

## Agile Software-Entwicklung und Microservices

Die klassische Methodik der Software-Entwicklung ist das Wasserfall-Modell. Man beginnt mit dem Schreiben eines Lastenhefts und der Definition der System-Architektur in der Entwurfsphase, gefolgt von der Implementierung, den Abnahmetests und final dem Ausrollen der Software. Bei einfachen Projekten mit klaren Anforderungen ergibt sich daraus eine große Planungssicherheit. In einem herausfordernden Projekt werden Probleme jedoch erst sehr spät sichtbar, da man erst zum Ende der Implementierung ein lauffähiges und testbares Produkt erhält. Anpassungen sind zu diesem Zeitpunkt schwierig und teuer, da

der gesamte Planungsaufwand bereits zu Projektbeginn erfolgt ist. In der Praxis benötigt man nach der Wasserfall-Methode mehrere Monate oder sogar Jahre, um ein neues Software-Release zu erstellen.

In Innovationsprojekten sind die Anforderungen zu Projektbeginn oft unklar. Man weiß beispielsweise noch nicht, wie hoch die Akzeptanz der Anwender ist oder ob sich das neue Geschäftsmodell in der Praxis bewährt. Gleiches gilt für die Leistungsfähigkeit neuer Technologien. Erst nach mehrmonatiger Entwicklung grundsätzliche Probleme, beispielsweise mit der Performance, zu erkennen, wäre fatal.

Ein alternativer Ansatz ist eine modulare Entwicklung, die Implementierung sogenannter „Microservices". Die Einzelmodule sind weniger komplex, schneller implementiert und leichter testbar. Bei agilen Projektmethoden wie SCRUM entwickelt man die einzelnen Module in Sprints von wenigen Tagen bis wenigen Wochen Dauer, in kleinen Teams, parallel und inkrementell. Jedes Inkrement führt zu einem isoliert lauffähigen und testbaren Produkt. Man überlässt den Teams die Auswahl der geeigneten Werkzeuge und Technologien, die sich von Modul zu Modul unterscheiden können. In täglichen Meetings und regelmäßigen Reviews werden Erfahrungen geteilt und Probleme schnell sichtbar. Umso wichtiger wird es aber, Routine-Aufgaben wie Kompilieren und Ausbringen von Software-Modulen vollständig zu automatisieren.

## DevOps als neue Unternehmenskultur

Schnell und kontinuierlich neue Software-Versionen auszubringen, ist eine sehr verantwortungsvolle Aufgabe, die nur funktionieren kann, wenn Entwickler und Administratoren gemeinsam Verantwortung übernehmen. Der Entwickler verantwortet seinen Code und seine Werkzeuge bis in die Produktion. Der Administrator arbeitet eng mit den Entwicklungsteams zusammen, um eine vollautomatisierte, nahtlose Ausbringung zu gewährleisten. Man braucht eine gemeinsame, gesunde Fehlerkultur.

Bei moderner Microservice-Entwicklung konzentriert man sich auf die zu implementierende Funktionalität und nicht auf Technologie-Schichten wie bei einer serviceorientierten Architektur oder einer Mehrschicht-Implementierung. Es gilt der Grundsatz: Mache genau eine Sache, aber mache sie richtig. Der Entwickler genießt große Freiheitsgrade. Er wählt für jedes Modul die seiner Meinung nach am besten geeigneten Werkzeuge, Programmiersprachen, Datenbanken und Laufzeit-Umgebungen. Die Kunst besteht darin, diese Freiheiten nicht zu einem nicht mehr handhabbaren Wildwuchs werden zu lassen.

Spätestens beim Datenschutz und bei der Datensicherheit hört der Spaß mit der Freiheit auf. Hier kommen neue Rollen auf Mitarbeiter mit Betriebserfahrung und Administratoren zu, die bei der agilen Entwicklung ansonsten oft als Blockierer gesehen werden. Sie können die agilen Teams bei der Automatisierung und auch in Sachen Daten-Sicherung, -Wiederherstellung und -Konsistenz ideal beratend unterstützen. Vor allem in der Umsetzung der EU-Datenschutz-Grundverordnung ist ein umsichtiger und vorausschauender Umgang mit Daten zwingend, um hohe Aufwände bei der Lösch- und Auskunftspflicht und schlimmstenfalls Strafen zu vermeiden.

Ein ideales agiles Team wäre also im Sinne der neuen Kultur eine heterogene Mischung aus Entwicklern und aus den Entwicklungsprozess beratend begleitenden Administratoren. Viele sich wiederholende Routinetätigkeiten aus dem Bereich der Datenbank- und System-Administration werden in Zeiten der Cloud vom Betreiber oder von autonomen Systemen erledigt. Der moderne Administrator wird sich in Zukunft qualitativen Aufgaben zuwenden können und die Rolle des vertrauenswürdigen Beraters in den Entwicklungsteams übernehmen.

## Klassische DevOps-Werkzeuge

Es gibt eine Vielzahl von Werkzeugen, die bei der Automatisierung des kompletten DevOps-Zyklus unterstützen; zumeist sind diese quelloffen. Der Oracle Developer Cloud Service *[2]* umfasst eine Sammlung der verbreitetsten Werkzeuge und deckt den kompletten Zyklus in Form eines Dienstes in der Oracle-Cloud ab. Dieser ist kostenfrei für Nutzer der Oracle-Universal-Credit-PaaS- und -IaaS-Dienste.

Die entsprechenden Werkzeuge lassen sich jedoch auch von Hand zusammenstellen. Ein DevOps-Zyklus startet mit der Projekt-Planung und -Verwaltung. „Jira" *[3]* ist ein gutes Beispiel für die Unterstützung agiler Entwicklung. Für das Code- und Versions-Management setzt man heute meist auf das von Linus Torwalds initiierte und verteilt nutzbare „Git". Der automatische Bau des Kompilats und die Zusammenstellung der Bibliotheken erfolgt mit „Maven", „Ant", „Grunt" oder „Gradle". Für die Ablaufkontrolle und das Ausbringen der Softwarepakete verwendet man „Hudson" oder „Jenkins". Eine Test-Automatisierung kann mit „JUnit" oder „Selenium" erfolgen.

Bei aller Automatisierung kommt dieses Prinzip bei der Microservice-Architektur an seine Grenzen. Jeder Microservice benötigt neben dem funktionalen Code auch eine Sammlung von Laufzeit-Umgebungen und Bibliotheken. Er sollte isoliert betreibbar sein, was das parallele Ausbringen auf geteilte Laufzeitinstanzen praktisch ausschließt. Man benötigt mehr als nur die automatische Provisionierung einer Applikation. Es geht auch um die massenhafte und vollautomatische Provisionierung von mehr oder minder komplexen Laufzeitumgebungen und Infrastrukturen.

## Container-basiertes DevOps für Microservices

Die Lösung für das Problem der isolierten Provisionierung von Microservices liegt in der Nutzung von Containern. Docker hat sich mittlerweile als Quasi-Standard für Container etabliert. Man packt alle benötigten Komponenten eines Microservice in einen gemeinsamen Docker-Container und betrachtet diesen als Ausbringungseinheit.

Für ein Container-basiertes DevOps sind drei Basis-Komponenten erforderlich: Erstens ein Werkzeug für die Erstellung und Verwaltung von Containergruppen; man spricht in diesem Zusammenhang auch von der Orchestrierung von Containern. Hier hat sich das ursprünglich von Google entwickelte und mittlerweile unter dem Dach der CNCF als Open-Source-Projekt weitergeführte Kubernetes (K8s) *[4]* gegenüber alternativen Ansätzen wie Docker Swarm oder Mesosphere durch-

gesetzt. Für die Ablage und Verteilung von Docker-Containern nutzt man private oder öffentliche Container Registries. Eine mit diesen integrierte Automatisierung der Erstellung und Ausbringung der Container ist die dritte Komponente. In Summe erhält man eine „Continuous Integration/ Continuous Delivery"-Pipeline (CI/CD) für Microservices auf Basis von Docker-Containern.

## Oracle Container Native Application Development Platform

Die Oracle Container Native Application Platform *[5]* bietet DevOps-Teams eine durchgängige Produktsammlung auf Open-Source-Basis, um Container-Cluster automatisiert aufsetzen und verwalten zu können. Die Container laufen virtualisiert, „Bare Metal", also ohne Hypervisor, oder in Kombination auf Basis der derzeit modernsten Oracle-Cloud-Infrastruktur (OCI). Diese ist etwa in der Oracle-Rechenzentrumsregion Frankfurt am Main verfügbar. Sie bietet mit ihren dediziert zuweisbaren Ressourcen höchste Performance, Verfügbarkeit und Ausfallsicherheit für den professionellen Unternehmenseinsatz. Die Oracle-Container-Plattform *(siehe Abbildung 1)* deckt den kompletten Lebenszyklus von Microservices ab:

- *Oracle Container Pipelines*
  Eine elegante, flexible und programmatische CI/CD Lösung, um Container-basierte Microservices schnell und wiederholbar auszuliefern
- *Oracle Container Registry (in Kürze verfügbar)*
  Eine hochverfügbare, private Container Registry, um Container-Images zu speichern und zu verteilen
- *Oracle Container Engine (in Kürze verfügbar)*
  Ein vollständig verwalteter Kubernetes-Dienst, der den Betrieb hochverfügbarer Cluster ermöglicht, ohne sich selbst um die Wartung und Administration von Kubernetes kümmern zu müssen

## Oracle Container Pipelines

„Bringe deine Software frühzeitig aus und bringe sie oft aus" ist das Motto der Microservice-Entwicklung. Die Oracle-Container-Plattform kann in diesem Sinne mit beliebigen CI/CD-Lösungen wie Jenkins oder Hudson genutzt werden. Container Pipelines sind die Container-zentrische CI/CD-Lösung von Oracle, basierend auf der Akquise von Wercker. Die Modellierung der Software-Auslieferung erfolgt hier in sogenannten „Workflow Pipelines". Ein Workflow startet automatisch, etwa mit einem „Git Push". Wercker kompiliert daraufhin die benötigten Code-Basen und führt alle notwendigen Schritte aus, das komplette Microservice-Projekt zu bauen. Es nutzt isolierte Container für die Code-Ausführung und generiert als Ergebnis des Workflows ebenfalls einen Docker-Container. Dieser enthält alle benötigten Konfigurationen und Umgebungen, um eigenständig und isoliert lauffähig zu sein. Er lässt sich in Container Registries ablegen und auf ein Ziel der Wahl, etwa die Oracle Container Engine, ausbringen.

## Oracle Container Registry

Bei der Container Registry (in Kürze verfügbar) handelt es sich um einen privaten, Docker-Registry-V2-konformen Ablageort für Images in der Oracle-Cloud. Sie ist vollständig mit der Container Pipeline und der Container Engine der Plattform integriert. So lassen sich mit der Container Pipeline generierte Images als Arbeitsschritt direkt und mit sehr geringer Latenz, hochverfügbar und sicher ablegen. Man kann die Registry durch Nutzung der Docker-CLI auch eigenständig vom Entwickler-Laptop, dem eigenem Rechenzentrum oder Cloud-Diensten von Dritt-Anbietern aus nutzen. Als Anwender bezahlt man übrigens nicht die Registry selbst, sondern nur die genutzten IaaS-Ressourcen der Oracle-Cloud.

## Oracle Container Engine

Das Open-Source-Werkzeug Kubernetes *[4]* automatisiert den Umgang mit Docker-Clustern vollständig. Dazu gehört die automatisierte Ausbringung, Skalierung, Lastverteilung und Speichernutzung. Das Werkzeug ist allerdings entsprechend seinen Aufgaben komplex in der Handhabung und bei einer Installation von Hand selbst zu betreuen und zu warten. Aus diesem Grund bietet Oracle ein über das CNCF-Konformitätsprogramm zertifiziertes Kubernetes als vollständig verwalteten und hochverfügbaren Dienst integ-

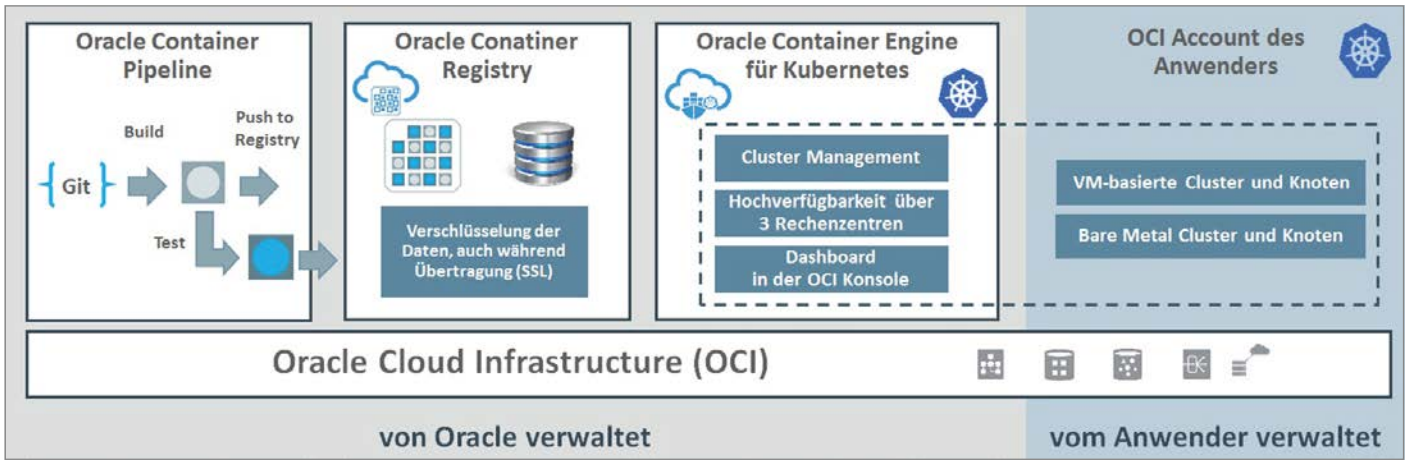



*Abbildung 1: Die Oracle-Container-Plattform*

riert mit der Oracle-Cloud-Infrastruktur (OCI). Man muss sich also nicht selbst um die Wartung der Kubernetes-Infrastruktur kümmern. Auch hier zahlt man als Anwender nicht den Kubernetes-Dienst selbst, sondern wiederum nur die genutzten IaaS-Ressourcen (wie Compute, Storage und Load Balancer).

Die von der Oracle Container Engine (in Kürze verfügbar) verwalteten Kubernetes-Cluster lassen sich flexibel erstellen und orchestrieren; entweder direkt aus der Container Pipeline heraus, über ein REST-API oder über das Benutzer-Interface der Oracle-Cloud. Die einzelnen Knoten sind per SSH erreichbar, damit man bei Bedarf weiterhin die gewohnten Docker-Werkzeuge nutzen kann. Die Zugriffsrechte des ganzen Teams lassen sich von der Entwicklung bis in die Produktion zentral verwalten. Die Vorteile der zugrunde liegenden Oracle-Cloud-Infrastruktur kommen voll zum Tragen: Man kann auf Knopfdruck Bare-Meta- Cluster mit voller vorhersagbarer Leistung über mehrere Verfügbarkeits-Domänen, also drei unabhängige Rechenzentren in einer Region, spannen.

## Serverless Computing mit Fn Project

Als die nächste Evolutionsstufe der modernen Applikationsentwicklung wird das Serverless Computing betrachtet. Man macht sich keine Gedanken mehr über Server, Storage, Netzwerk, Virtualisierung und andere Infrastruktur-Komponenten, sondern konzentriert sich ganz auf den eigenen Code, also die Umsetzung der Geschäftsanforderung in Form eines kleinen Funktionspakets (Function-as-a-Service, FaaS). Server gibt es beim Serverless Computing natürlich dennoch. Sie brauchen allerdings keinerlei Administration und bleiben für den Entwickler unsichtbar.

Mit AWS Lambda *[6]*, Cloud Functions *[7]* und anderen gibt es schon einige kommerzielle, weitgehend geschlossene Serverless-Computing-Angebote am Markt. Mit Fn Project *[8]* hat Oracle eine Open-Source-Serverless-Plattform als Alternative initiiert. Im Gegensatz zu den vorgenannten Plattformen nutzt man bei Fn Project nicht nur ein API oder eine Web-Oberfläche, um Code hochzuladen.

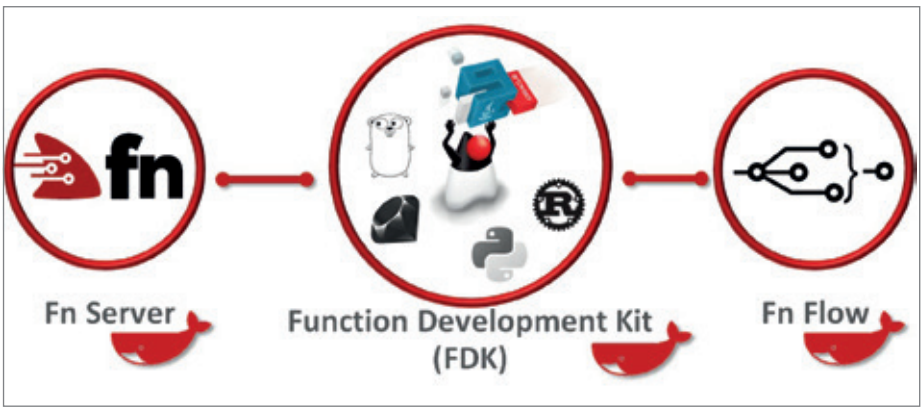



*Abbildung 2: Der Aufbau von Fn Project*

Fn Project ist komplett Container-basiert *(siehe Abbildung 2)*. Die Funktion ist hier der Container und der Container die Funktion, Docker die einzige Abhängigkeit. Selbst der Fn Server, die Laufzeit-Umgebung von Fn Project, ist ein Docker-Container. Dadurch ist er unverändert überall lauffähig: auf dem Laptop des Entwicklers, auf jeder beliebigen Container-Plattform im eigenen Rechenzentrum oder als Kubernetes-Cluster in der Oracle-Cloud-Infrastruktur (OCI).

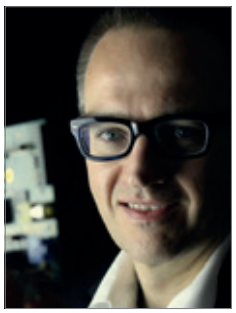
Der Fokus von Fn Project liegt auf der einfachen und flexiblen Nutzbarkeit. Es ist polyglott, Funktionen können also in jeder beliebigen Programmiersprache erstellt werden. Es gibt bereits vorgefertigte Entwicklungsbaukästen (FDK, Function Development Kit) für Java, Go, JavaScript, Ruby, Rust und Python. Man entwickelt von der Kommandozeile aus mit dem Kommando „fn“. Funktionen werden als Docker-Container gepackt, die auch die Ausbringungseinheit sind. Mit einem vorgefertigten Helm-Chart lässt sich Fn Project als Kubernetes-Cluster automatisiert in der Oracle-Cloud aufsetzen und passt sich damit bestens in die Oracle-Container-Plattform ein.

Mit Fn Flow lassen sich einzelne Funktionen zu komplexen Diensten zusammensetzen. Dabei werden Aspekte wie Parallelisierung, Ausführungs-Reihenfolge, Wiederanläufe und Fehlerbehandlung adressiert. Es ist im Vergleich zu einem klassischen Service-Bus oder einer SOA-Orchestrierung ein Leichtgewicht. Abläufe werden über ein asynchrones und verteiltes Programmiermodell im Code definiert. Ausgeführt wird Fn Flow ebenfalls als Funktion.

**Hinweis:** Oracle Container Registry und Oracle Container Engine sind in Kürze verfügbar, Stand 04/18

## Weitere Informationen

[1] *https://cloud.oracle.com/tryit*
[2] *https://cloud.oracle.com/developer-service*
[3] *https://www.atlassian.com/software/jira*
[4] *https://kubernetes.io*
[5] *https://cloud.oracle.com/containers*
[6] *https://aws.amazon.com/lambda*
[7] *https://cloud.google.com/serverless/functions*
[8] *https://fnproject.io*



Marcel Amende
marcel.amende@oracle.com