



TECHNISCHE HOCHSCHULE NÜRNBERG
GEORG SIMON OHM

Fakultät Informatik

**Automatisierte
Provisionierungsmechanismen für
Laufzeitumgebungen von Legacy z/OS
Anwendungen mit „IBM Cloud
Provisioning and Management for z/OS“
am Beispiel der „Rechnungsschreibung“
bei DATEV e.G.**

Bachelorarbeit im Studiengang Informatik

vorgelegt von

David Krug

Matrikelnummer 3036355

Erstgutachter: Prof. Dr. Korbinian Riedhammer

Zweitgutachter: Prof. Dr. Friedhelm Stappert

Dieses Werk einschließlich seiner Teile ist **urheberrechtlich geschützt**. Jede Verwertung außerhalb der engen Grenzen des Urheberrechtsgesetzes ist ohne Zustimmung des Autors unzulässig und strafbar. Das gilt insbesondere für Vervielfältigungen, Übersetzungen, Mikroverfilmungen sowie die Einspeicherung und Verarbeitung in elektronischen Systemen.

Prüfungsrechtliche Erklärung der/des Studierenden

Angaben des bzw. der Studierenden:

Name: _____ Vorname: _____ Matrikel-Nr.: _____

Fakultät: _____ Studiengang: _____

Semester: _____

Titel der Abschlussarbeit:

Ich versichere, dass ich die Arbeit selbständig verfasst, nicht anderweitig für Prüfungszwecke vorgelegt, alle benutzten Quellen und Hilfsmittel angegeben sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

Ort, Datum, Unterschrift Studierende/Studierender

Erklärung zur Veröffentlichung der vorstehend bezeichneten Abschlussarbeit

Die Entscheidung über die vollständige oder auszugsweise Veröffentlichung der Abschlussarbeit liegt grundsätzlich erst einmal allein in der Zuständigkeit der/des studentischen Verfasserin/Verfassers. Nach dem Urheberrechtsgesetz (UrhG) erwirbt die Verfasserin/der Verfasser einer Abschlussarbeit mit Anfertigung ihrer/seiner Arbeit das alleinige Urheberrecht und grundsätzlich auch die hieraus resultierenden Nutzungsrechte wie z.B. Erstveröffentlichung (§ 12 UrhG), Verbreitung (§ 17 UrhG), Vervielfältigung (§ 16 UrhG), Online-Nutzung usw., also alle Rechte, die die nicht-kommerzielle oder kommerzielle Verwertung betreffen.

Die Hochschule und deren Beschäftigte werden Abschlussarbeiten oder Teile davon nicht ohne Zustimmung der/des studentischen Verfasserin/Verfassers veröffentlichen, insbesondere nicht öffentlich zugänglich in die Bibliothek der Hochschule einstellen.

Hiermit ☐ genehmige ich, wenn und soweit keine entgegenstehenden
Vereinbarungen mit Dritten getroffen worden sind,
☐ genehmige ich nicht,

dass die oben genannte Abschlussarbeit durch die Technische Hochschule Nürnberg Georg Simon Ohm, ggf. nach Ablauf einer mittels eines auf der Abschlussarbeit aufgebrachten Sperrvermerks kenntlich gemachten Sperrfrist

von _____ Jahren (0 - 5 Jahren ab Datum der Abgabe der Arbeit),

der Öffentlichkeit zugänglich gemacht wird. Im Falle der Genehmigung erfolgt diese unwiderruflich; hierzu wird der Abschlussarbeit ein Exemplar im digitalisierten PDF-Format auf einem Datenträger beigelegt. Bestimmungen der jeweils geltenden Studien- und Prüfungsordnung über Art und Umfang der im Rahmen der Arbeit abzugebenden Exemplare und Materialien werden hierdurch nicht berührt.

Ort, Datum, Unterschrift Studierende/Studierender

Kurzdarstellung

Innerhalb der DATEV eG gewinnen PaaS (Plattform as a Service) Ansätze immer mehr an Bedeutung. Ein Vorteil dabei ist die unkomplizierte, automatisierte Provisionierung von Laufzeitumgebungen. Im Vergleich dazu ist der Bereitstellungsprozess für legacy z/OS Anwendungen mit vielen manuellen Schritten und vielen Absprachen, auch abteilungsübergreifend, verbunden.

Das Ziel dieser Forschung ist es zu bestimmen, ob die automatische Bereitstellung von Laufzeitumgebungen für legacy z/OS Anwendungen möglich ist. Dazu werden folgende Forschungsfragen gestellt:

1. Ist es technisch möglich mit dem „IBM Cloud Provisioning and Management for z/OS“-Toolkit eine Laufzeitumgebung für legacy z/OS Anwendungen automatisiert bereitzustellen?
2. Wird dadurch der aktuelle Bereitstellungsprozess schneller und auch sicherer?
3. Erzeugt die Nutzung einen Mehrwert bei den Stakeholdern, also den Entwicklerteam und den Administratorenteams?

Um diese Forschungsfragen zu beantworten, wurde zunächst anhand einer Beispielanwendung, der DATEV Rechnungsschreibung, das „IBM Cloud Provisioning and Management for z/OS“-Toolkit untersucht. Das Toolkit bietet zwei Möglichkeiten für die automatisierte Bereitstellung von Laufzeitumgebungen. Es wurde sich für eine dieser Möglichkeiten entschieden, diese wurde implementiert.

Anschließend wurde der dadurch ermöglichte Prozess aufgezeigt und mit dem etablierter Prozess verglichen. Dabei stellte sich heraus, dass der ermöglichte Bereitstellungsprozess schneller und durch weniger Absprachen auch weniger fehleranfällig ist. Dennoch ist die Lösung nicht optimal und das Toolkit bietet weitere Möglichkeiten zur Verbesserung.

Schließlich sind Interviews mit den Stakeholdern bezüglich eines Mehrwertes des neuen Prozesses durchgeführt worden. Sowohl die befragten Entwickler als auch die befragten Administratoren sehen in dem Toolkit eine Chance auf Verbesserung des aktuell etablierter Bereitstellungsprozesses. Jedoch ist die momentane Lösung zwar funktionsfähig, aber noch bezüglich firmenweiten und einfacheren Einsatz zu optimieren.

Auf dieser Grundlage lässt sich sagen, dass das „IBM Cloud Provisioning and Management for z/OS“-Toolkit die automatisierte Bereitstellung von Laufzeitumgebungen für legacy z/OS Anwendungen ermöglicht. Die in dieser Arbeit implementierte Möglichkeit ist nicht optimal, bietet aber bereits einen Mehrwert für die Stakeholder. Weiterführende Forschung könnte sich mit der Implementierung der zweiten Möglichkeit und mit den damit verbundenen weiteren Verbesserungen des Bereitstellungsprozesses beschäftigen.

Inhaltsverzeichnis

1. Einleitung	1
1.1. Problemstellung	3
1.2. Ziel der Arbeit	5
2. Grundlagen	7
2.1. cloud-native bei DATEV e.G.	7
2.1.1. „Cloud Foundry“	8
2.1.2. „CI/CD-Pipeline“	8
2.2. Mainframe / Großrechner	10
2.3. Mainframe Anwendungen bei DATEV e.G.	11
2.4. Subsysteme / Middleware	11
2.4.1. Customer Information Control System	12
2.4.2. Db2	13
2.4.3. IBM MQ	13
2.5. „IBM Cloud Provisioning and Management for z/OS“	15
2.5.1. z/OS Management Facility	18
2.5.2. z/OS Provisioning Toolkit	19
3. Vorgehensweise	21
4. Analyse	25
4.1. Analyse von cloud native bei DATEV e.G.	25
4.2. Aktueller Bereitstellungsprozess	26
4.2.1. Bereitstellung einer CICS Instanz	27
4.2.2. Bereitstellungsprozess einer Db2 Datenbank	29
4.2.3. Bereitstellungsprozess einer IBM MQ Queue	31
4.2.4. Zusammenfassung aktueller Bereitstellungsprozess	31
4.3. DATEV-Rechnungsschreibung	33
4.3.1. Tägliche Bewertung	34
4.3.2. Preisermittlung	34
5. Realisierung	37
5.1. Vergleich zwischen z/OSPT und z/OSMF	37

5.2. Testplex	38
5.2.1. „cics_getting_started“-Template	39
5.2.2. „cics_54“-Template	39
5.3. Entwicklungsstages	47
5.3.1. CICS Anpassung	47
5.3.2. Db2 Anpassung	48
5.3.3. IBM MQ Anpassung	48
5.3.4. Testablauf	50
5.4. Bereitstellungsprozess aktuelles Template	51
5.4.1. Use-Case: Neue Template Instanz	51
5.4.2. Use-Case: Zusätzliche Template Instanz	51
5.4.3. Use-Case: Änderungen durch Administratorenteam	52
5.5. Fazit Realisierung	52
5.6. Interviews	55
5.6.1. CICS Administratoren	55
5.6.2. Db2 Administratoren	55
5.6.3. Meinungsbild	57
6. Ausblick	59
7. Zusammenfassung	63
A. Anhang	65
A.1. Agenda der neunzehnten Academic Mainframe Consortium e.V. Tagung vom 16.01.2020 bis 17.01.2020	65
A.2. IT workload distribution worldwide in 2018 and 2020, by cloud type	68
A.3. Produktstammdaten Tabellen Data Definition Language	68
A.4. Interview Fragebögen	84
A.5. Workflow Step mit REST-Call	94
Abbildungsverzeichnis	95
Tabellenverzeichnis	97
Quellcodeverzeichnis	99
Literaturverzeichnis	101

Kapitel 1.

Einleitung

„I recently predicted the last mainframe will be unplugged on March 15, 1996“¹ - ein in der Großrechner-Welt bekannt gewordenes Zitat. Es handelt sich um eine 1993 getroffene Vorhersage, nämlich dass der letzte Mainframe, auch Großrechner genannt, am 15 März 1996 abgeschaltet werden würde. Warum war diese Vorhersage falsch? Wieso wird sich im Jahre 2020 immer noch mit dieser Technologie beschäftigt? Und was genau ist ein Großrechner?

Kurz gesagt ist ein Großrechner² ein leistungsstarkes, zentralisiertes Serversystem. In dieser Arbeit wird nur auf Mainframes aus dem Hause IBM, die sogenannte z-Plattform, eingegangen. Damit ist auch der Technologiestack festgelegt. Das verwendete Betriebssystem ist z/OS, darauf werden Middleware Produkte wie CICS³, das Datenbanksystem Db2⁴ sowie die Messaging Lösung „IBM MQ“⁵ betrieben. Als Programmiersprachen werden z.B. COBOL, IBM Assembler, C und C++ verwendet. Seit ca. 1997 ist auch Java auf dem Mainframe verfügbar.⁶

Der IBM Mainframe hat eine lange Geschichte. Vor mehr als fünfzig Jahren wurde der erste Großrechner, das sog. „System/360“ vorgestellt. Bis in die 90er Jahre spielte der IBM Mainframe eine Hauptrolle auf dem Computermarkt, dann gewannen zunehmend verteilte Client-Server-Systeme an Bedeutung.⁷ Seitdem gilt der Mainframe bereits als „legacy“ und damit als „Altlast“⁸.

Wieso also wird sich mit der Mainframe Technologie noch beschäftigt? Eine Antwort: Auf dem Mainframe werden auch im Jahr 2020 geschäftskritische Anwendungen in der ganzen Welt gehostet. So verwenden laut IBM 92 der 100 weltweit führenden Banken für ihre

¹[[Also 93](#)]

²Beschreibung im Absatz 2.2 zu finden

³Anwendungsserver, CICS Beschreibung Absatz 2.4.1

⁴Beschreibung Absatz 2.4.2

⁵Beschreibung im Absatz 2.4.3 zu finden

⁶[[Stee 03](#)]

⁷[[Ceru 03](#)]

⁸[[http 20i](#)]

Kernabläufe einen IBM Mainframe. Dies beinhaltet 87 Prozent aller Kreditkartentransaktionen und ca. 350.000 Transaktionen pro Sekunde.⁹ Inklusiver dieser Transaktionen verarbeiten Großrechner heutzutage weltweit circa 1,2 Millionen CICS Transaktionen pro Sekunde.¹⁰ Im Vergleich hierzu werden 63.000 Google Suchanfragen pro Sekunde abgesetzt.¹¹

Aus der Kombination von hohem Workload, der Abhängigkeit von einem Hersteller (IBM) und dem als veraltet geltenden Technologiestack entstehen jedoch zunehmend Risiken. Es wird immer schwieriger, Nachwuchs in diesem Bereich zu finden. Zum einem, da Mainframe-Know How kaum noch an Universitäten gelehrt wird. Die Seite des Hochschulkompass¹² liefert z.B. weder für „Mainframe“ noch für „Großrechner“ einen Treffer. Zum anderen ist der demographische Faktor bei den Wissensträgern nicht zu vernachlässigen. Diese sind - wie die Technologien auf dem Mainframe - in die Jahre gekommen und erreichen das Rentenalter.¹³

Ein weiteres Problem ist, dass eine Firma, die einen IBM Großrechner mit z/OS betreibt, von dem oben genannten proprietären Technologiestack abhängig ist, dass heißt, es existiert eine starke Hersteller- und Plattformabhängigkeit, z.B. in Bezug auf CICS, Db2, IBM-COBOL-Compiler, Assembler.

Offensichtlich betreiben dennoch etliche Firmen einen IBM Großrechner. Dazu zählen hauptsächlich Banken, Versicherungen, Fluggesellschaften usw.¹⁴ Der gemeinsame Nenner dieser Unternehmen ist, dass sich über die Jahre und Jahrzehnte enorme Investitionen auf dem Mainframe angesammelt haben. Die entstandenen, hochgradig geschäftskritischen Kernsysteme haben hohe Anforderungen an Massendatenverarbeitung, Sicherheitsstandards und Hochverfügbarkeit. All diese Punkte sprechen nach wie vor für die Nutzung eines Großrechners, z.B. auch bei der DATEV e.G. (Kommentar: hier wäre ein Zitat schön, von einer Bank o.ä. ich schau mal ob ich was finde)

Die DATEV e.G. wurde am 14.02.1966 von 65 Steuerbevollmächtigten gegründet. Sie verfolgten mit der Gründung das Ziel, Buchführungsaufgaben für ihre Mandanten mit Hilfe der neu aufkommenden EDV zu bewältigen. Aufgrund hohen Mitgliederwachstums wurde hierfür bereits 1969 in einen firmeneigenen IBM-Großrechner investiert.^[http 19b] Heute umfasst das Leistungsspektrum der DATEV e.G. unter anderem das Rechnungswesen, Personalwirtschaft, Consulting, IT-Sicherheit, Weiterbildung für ihre Kunden, in erster Linie Steuerberater, Wirtschaftsprüfer und Rechtsanwälte, und deren Mandanten. Ein nicht unbeachtlicher Teil dieser betriebswirtschaftlichen Anwendungen läuft bis heute ganz oder als Backend von Client-Anwendungen auf einem IBM Großrechner im DATEV Rechenzentrum.

⁹^[http 20j]

¹⁰^[http 19c]

¹¹^[http 19a]

¹²^[http 20k]

¹³^[http 20e]

¹⁴^[http 20j]

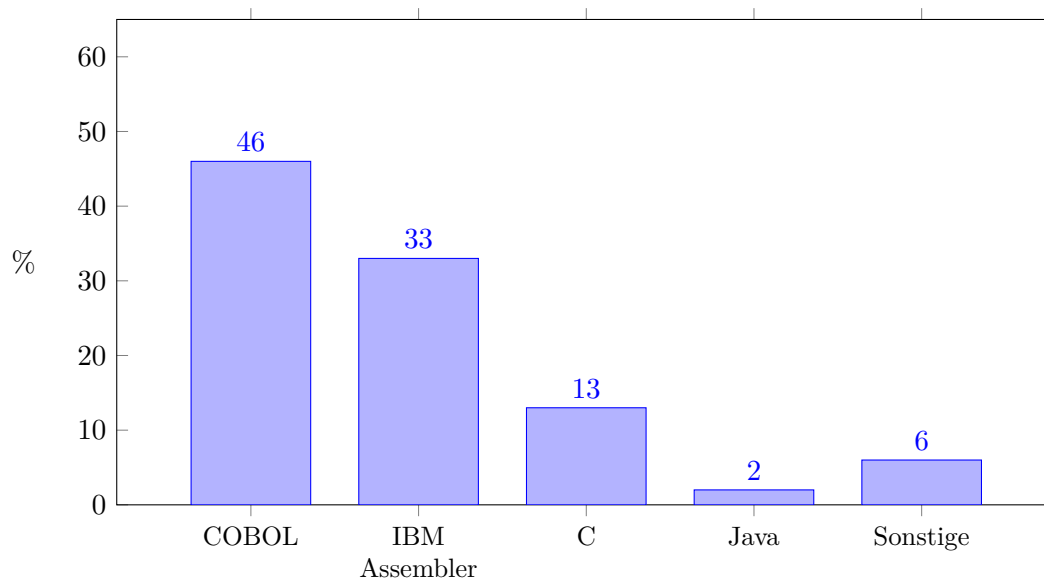


Abbildung 1.1.: Anteil der verwendeten Programmiersprachen auf dem Mainframe bei DATEV eG in Prozent

So werden pro Tag circa 150.000 Batch Jobs¹⁵ und circa 90 Millionen CICS-Transaktionen verarbeitet. Diese Last wird von circa 14.000 aktiven Modulen erzeugt. Wie in der Abbildung 1.1 zu sehen ist, ist COBOL mit circa 46% Prozent die am häufigsten verwendete Programmiersprache am Großrechner bei der DATEV e.G.. Durch diese Module werden unter anderem im Monat circa 11 Millionen Lohnabrechnungen erstellt und circa eine Millionen Umsatzsteuer-Voranmeldungen durchgeführt. 2018 wurde mit den DATEV Produkten erstmals die Umsatz-Milliarde erreicht.¹⁶

1.1. Problemstellung

Im Jahre 2020 ist der größte Konkurrent für den Mainframe die Cloud. Laut einer Vorhersage aus dem Jahr 2018¹⁷ soll im Jahre 2020 circa 79 Prozent des weltweiten Workloads in einer Cloud verarbeitet werden. Für die Entwicklung von neuen Online-Anwendungen im cloud-native Stil wurde bei der DATEV e.G. eine Platform-as-a-Service (PaaS)-Lösung geschaffen und neue DevOps¹⁸ Prozesse aufgebaut. Dies ist im Cloud-Zeitalter nötig, um mit einer verbesserten Entwicklungseffizienz und neuen Architekturen Anwendungen (Äpps") schneller auf den Markt bringen und auf Kundenanforderungen schneller reagieren zu können. Stichwort: Continuous Integration, Continuous Delivery (CI/CD). Ein Baustein der effizienteren Prozesse durch PaaS sind die sog. „Self Services“. D.h., Entwicklerteams können sich über

¹⁵Beschreibung in Absatz ??

¹⁶[[http 20h](#)]

¹⁷Statistik im Anhang A.1

¹⁸Siehe Absatz 4.1

sog. „Cloud Services“, z.B. Datenbanken wie PostgreSQL, Mongo und Messaginglösungen wie Kafka entweder manuell über einen Marktplatz (siehe Abbildung ??) oder automatisiert per „Build-Pipeline“ eine Laufzeitumgebung für ihre Anwendung zusammenbauen. Eine genaue Beschreibung der Begrifflichkeiten erfolgt im Absatz 2.1. Den Entwicklern steht, neben modernen Entwicklungsumgebungen (IDE¹⁹) und einer Sourceverwaltung mit GIT, auch eine sog. „Toolchain“ zur Verfügung. Diese beinhaltet Tools für Build, Test, Quality Gates und Deployment. Damit wird der Entwicklungsprozess automatisiert und man erhofft sich eine hohe Entwicklereffizienz. (Kommentar: hier würde ich nach wie vor die Grafik von unserem go/cloud Sharepoint mit einbinden)

Der Entwicklungsprozess für z/OS Anwendungen bei DATEV e.G. erscheint im Vergleich zu dieser PaaS-Lösung veraltet. So wurde 2010 eine auf Eclipse basierende Entwicklungsumgebung für COBOL und IBM Assembler in der DATEV e.G. flächendeckend bereitgestellt. Zuvor - und teilweise heute noch - wurde mit Hilfe der in Abbildung 1.2 gezeigten Oberfläche, dem sog. ISPF gearbeitet. Diese stellte z.B. nur ein Syntaxhighlighting zur Verfügung. Ein Meilenstein für die modernisierte z/OS Entwicklung bei DATEV war die Einführung

```

000052  if datatype(anzahl,'N') then do
000053      do ix = 1 to k
000054          say csqout.ix
000055      end
000056  end
000057  /* interaktiv ? */
000058  else
000059      address $datev "browse stem csqout. 200"
000060
000061  Exit rcce
000062  /* ----- */
000063  Generate:
000064  /* ----- */
000065  /* Template abfragen
000066  /* ----- */
000067  rcg = inq_template()
000068  if rcg > 0 then return rcg
000069
000070  Do i=1 to Loop
000071
000072      q_new= q_hlq!!Right(1,6,'0')
000073      if template_qtype = 'QLOCAL' then
000074          command = "DEFINE REPLACE QL("q_new") LIKE("q_template")"
000075          "QSGDISP("template_qsgdisp")"
000076
000077      if template_qtype = 'QREMOTE' then
000078          command = "DEFINE REPLACE QR("q_new") LIKE("q_template")"
000079          "QSGDISP("template_qsgdisp") RNAME("q_new")"
000080
000081  /* ----- */
000082  /* alle 10 Schleifen kurz warten
000083  /* ----- */
000084  if 1//10 = 0 then
000085      Call Wait 1
000086      rcg = command_send()
000087      if rcg > 0 then leave
000088  End
000089  return rcg
000090  /* ----- */

```

Abbildung 1.2.: Auszug aus einem REXX Skript in der ISPF Oberfläche

von GIT im Jahr 2018 auch für z/OS Sourcen. Dieses weit verbreitete Standard-Tool ist im z/OS Umfeld tatsächlich eine entscheidende Neuerung. Dadurch wurde ein bis dato verwendetes eigenentwickeltes Tool für die Sourceverwaltung der z/OS Sourcen abgelöst. Dieses stellte nur ein sehr einfaches Versionierungskonzept ohne aus Git bekannte Features wie Merge, Branch usw. bereit. Parallelentwicklung von verschiedenen Features war vorher mit viel Aufwand und Abstimmung möglich. Das Tooling wurde somit modernisiert, jedoch nicht der Entwicklungsprozess selbst. Es teilen sich sehr viele Anwendungen die gleichen Entwicklungs-CICS/Db2/MQ Ressourcen. Das heißt auch, dass eine Parallelentwicklung - trotz jetzt möglichen Git-Branche - an unterschiedlichen Features nur mit viel Abstimmungsaufwand und Absprachen innerhalb eines Entwicklungsteams, teilweise

¹⁹Integrated Development Environment wie IntelliJ oder Eclipse

auch abteilungsübergreifend, möglich ist. Werden Änderungen an bestehenden Ressourcen durchgeführt oder werden neue Systemumgebungen benötigt, entsteht weiterer Abstimmungsaufwand und weitere Absprachen. Dadurch ist der aktuelle Prozess fehleranfällig und langsam.

Es bleibt die Frage, wie wird vor diesem Hintergrund mit den vielen Mainframebestandsanwendungen bei der DATEV e.G. in Zukunft umgegangen? Die komplette Ablösung dieser Anwendungen durch cloud-native Lösungen ist eine Option, deren zeitlicher Rahmen und Machbarkeit aktuell nicht absehbar ist.²⁰ Für die Funktionsfähigkeit dieses Bestandsgeschäfts, das die Core-Business-Funktionalitäten der DATEV e.G. darstellt, muss also effiziente Weiterentwicklung und Wartung gewährleistet werden. Auch im Falle einer geplanten Ablöse von Anwendungen muss je nach Strategie (z.B. „Rewrite“/ „Rearchitect“)²¹ das Alt-System parallel dazu über Jahre oder Jahrzehnte gepflegt und funktional aktuell gehalten werden. Daraus folgt, dass aus Sicht der DATEV e.G. weiter in die IBM Mainframe Plattform investiert werden muss. Dies bedeutet Investitionen in die bereitgestellte Infrastruktur (Hardware, Betriebssysteme, Lizenzen), insbesondere aber auch Investitionen, die die oben genannten Anforderungen an Weiterentwicklung, Wartung und Entwicklungseffizienz sowie Effizienz im Betrieb adressieren.

1.2. Ziel der Arbeit

Es liegt also nahe, sich an den oben beschriebenen Prozessen zu cloud native Entwicklung zu orientieren. In diesem Zusammenhang läuft aktuell bei DATEV e.G. ein Proof of Concept bezüglich automatisierter Builds von z/OS Anwendungen auf Basis von Jenkins basierten Pipelines. Dies ist auch die Voraussetzung für automatisierte Tests von z/OS Programmen im Rahmen des „Continuous Integration, Continuous Deployment“ Ansatzes. Was jedoch fehlt, sind „Self Services“ für Laufzeitumgebung und Middleware. Die dafür notwendige automatisierte Provisionierung einer z/OS Anwendungsumgebung, d.h Laufzeit, Middleware etc., ist aktuell noch weitgehend unerforscht. Hier sind die Prozesse bei DATEV und anderen Kunden oft noch proprietär, hoch spezialisiert, manuell und nicht modernisiert. Gerade bei Mitarbeitern im Betrieb, die als Administratoren für die Middleware-Produkte arbeiten, sind die Bedenken groß, ob man diese Cloud-Vorgehensweise auf hochspezialisierte individuelle Komponenten wie CICS, DB2, IBM MQ anwenden kann. IBM bietet eine Lösung mit dem „IBM Cloud Provisioning and Management for z/OS“-Toolkit. Dies hat sich noch

²⁰??

²¹(Kommentar, Strategiepatters lt. Gartner, ich schick Dir enien Link)

nicht flächendeckend durchgesetzt, aber es herrscht großes Interesse an Erfahrungen und Einschätzungen bei IBM Kunden. Hier setzt diese Arbeit an und klärt folgende Fragen:

- Ist es möglich, den Bereitstellungsprozess für z/OS Anwendung bei DATEV e.G. mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“-Tools an cloud native Prozesse anzunähern?
- Erzeugt die Nutzung von „IBM Cloud Provisioning and Management for z/OS“ einen Mehrwert bei den Stakeholdern, also den Entwicklerteams und den Administratorenteams?

Um diese Fragen zu beantworten wird die Provisionierung einer z/OS Laufzeitumgebung für eine bestehende Anwendung untersucht. Diese Anwendung sollte CICS als Anwendungs-server, eine Db2 Datenbank und IBM MQ als Messaginglösung nutzen, um für diese 3 Haupt-Technologien (Middleware-Komponenten) eine Aussage treffen zu können. Die genaue Vorgehensweise wird im Kapitel **3** beschrieben.

Kapitel 2.

Grundlagen

Um den Unterschied zwischen modernen cloud native Entwicklungsprozessen und dem Mainframe Entwicklungsprozess darstellen zu können, werden zunächst Begriffe aus dem cloud-native-Umfeld erläutert. Dabei wird auch auf die Vorteile und Begrifflichkeiten des damit ermöglichten Entwicklungsprozesses eingegangen. Anschließend werden für die Beantwortung der Forschungsfragen relevante Begriffe des Mainframe-Umfelds beleuchtet.

2.1. cloud-native bei DATEV e.G.

Eine cloud-native Anwendung ist eine speziell für das Cloud-Computing¹ konzipierte und entwickelte Anwendung. Oft werden damit Online-Anwendungen und mobile „Apps“ entwickelt, für die häufig und hoch frequent neue Features bereitgestellt werden sollen. Für solche Anwendungen ist das Architekturpattern der sogenannten „Microservices“, weit verbreitet. Diese einzelnen, entkoppelten Services sind beispielsweise in Containern paketierte. Bekannt geworden ist hier der Ansatz von Docker². Die Docker Container werden über Images beschrieben und beinhalten neben der Anwendung alles das, was die Anwendung an Komponenten zur Laufzeit benötigt, Bibliotheken, Dateien.³ Somit kann die Anwendung auf verschiedenen „privat“ und „public“ Cloud-Umgebungen, auch von unterschiedlichen Anbietern, ausgeführt werden. Große Anbieter sind hier beispielsweise AWS (Amazon), Google, Microsoft Azure.⁴ Dort können bereitstehende Services für Datenhaltung, Security, Messaging usw. genutzt werden. [[http 20g](#)]

Eine moderne cloud-native Anwendung innerhalb der DATEV e.G. machen folgende Dinge aus:

- „Cloud Foundry“
- CI/CD-Pipeline

¹Glossar ??

²Glossar ??

³[[Vohr 16](#)]

⁴[[http 20d](#)]

2.1.1. „Cloud Foundry“

Bei Cloud Foundry handelt es sich um eine quelloffene Platform-as-a-Service, kurz PaaS. Platform-as-a-Service, beschreibt neben Infrastructure-as-a-Service, kurz IaaS und Software-as-a-Service, kurz SaaS, einen Grad an Auslagerung von IT-Systemen in die Cloud. Im Vergleich zu SaaS, bei der ganze Anwendungen in einer Cloud zur Verfügung stehen, und IaaS, bei der die automatisierte Bereitstellung von Infrastrukturkomponenten wie Netzwerk, Speicher usw. im Fokus steht, stellt eine PaaS-Lösung eine Plattform, die sich neben der Infrastruktur auch um das Betriebssystem, die Middleware und die Laufzeitumgebung kümmert, bereit.⁵ Für die Verwaltung von Ressourcen bietet Cloud Foundry eine Weboberfläche, den sogenannten „Marketplace“, an. In diesem können mit wenigen Mausklicks Schnittstellen zu Services wie Datenbankmanagementsysteme, Messaging- oder Monitorlösungen zur Anwendung hinzugefügt werden. Diese Schnittstellen, auch „Self-Service“ oder „Service-Broker“ genannt, können mittels einer von Cloud Foundry zur Verfügung gestellten API selbst entwickelt werden. Daneben kümmert sich Cloud Foundry um das Staging der Anwendungen. D.h. eine Anwendung kann mit den benötigten Komponenten sicher von einer Entwicklungs- in eine QS- bzw. Produktiv-Stage verschoben werden. Die notwendigen stagespezifischen Anpassungen werden konfigurativ beigesteuert. Um eine Anwendung in einer bestimmten Stage bereitzustellen, bietet Cloud Foundry ein Kommandozeileninterface an. Neben der Bereitstellung können mit diesem Interface beispielsweise Anwendungen auch horizontal skaliert werden. Um die Bereitstellung über mehrere Stages hinweg zu automatisieren kommt bei der DATEV e.G. eine auf Jenkins basierende CI/CD-Pipeline zum Einsatz. [[http 20f](#)]

2.1.2. „CI/CD-Pipeline“

CI/CD steht für „Continuous Integration und Continuous Delivery“ in manchen Fällen auch für „Continuous Integration, Continuous Delivery und Continuous Deployment“. Die einzelnen Begriffe werden im Folgenden erläutert, als Überblick dient Abbildung 2.1

„Continuous Integration“

Continuous Integration beschreibt einen Prozess, bei dem Änderungen von Entwicklern regelmäßig in eine gemeinsame Codebasis integriert und getestet werden. In Abbildung 2.2 ist der Prozess dargestellt. Eine Voraussetzung für den Einsatz von CI ist eine zentrale Sourceverwaltung. Bei der DATEV e.G. handelt es sich dabei um GIT. Nachdem ein Entwickler Änderungen am Code vorgenommen hat, lädt er diese in die Sourceverwaltung hoch. Dabei kann eine erste Überprüfung des Codes mittels statischer Codeanalyse durchgeführt werden. Dazu zählt unter anderem die Prüfung vorher definierter Coderichtlinien.

⁵[?]

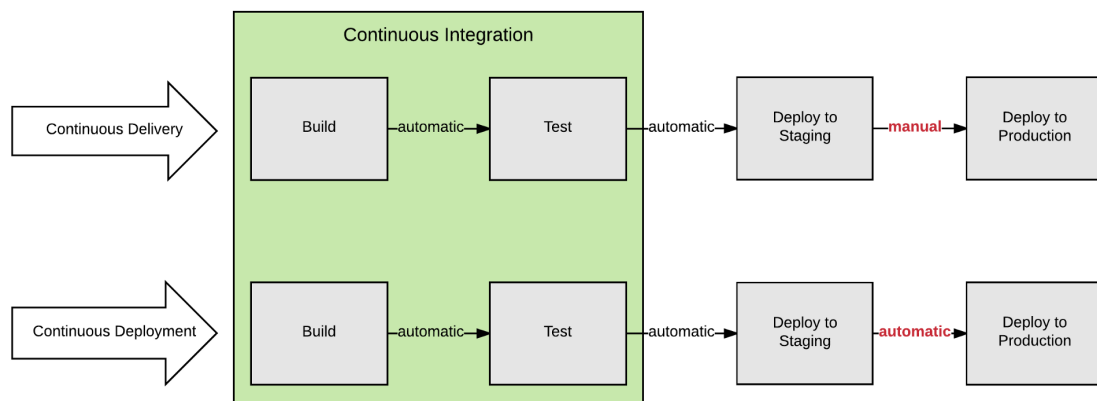


Abbildung 2.1.: Abgrenzung von Continuous Integration, Continuous Delivery und Continuous Deployment (Quelle: <http://20b>)

Sind diese erfolgreich, wird ein isolierter CI-Server benachrichtigt. Dieser Server holt sich den neusten Stand des Quellcodes und baut diesen, um anschließend automatisierte Tests durchzuführen. Hier liegt der zentrale Vorteil der Automatisierung. Bei jedem Bauen wird automatisch überprüft, ob das Ergebnis das erwartete ist. Voraussetzung dafür ist die oben beschriebene Paketierung der Anwendung mit der Laufzeitumgebung und den benötigten Komponenten.

Bei den Tests während der Continuous Integration handelt es sich um sogenannte „Unit-tests“. Dabei wird anhand vordefinierter Eingangsdaten die Ausgabe bestimmter Funktionen geprüft (Soll-Ist-Vergleich). Dazu gehört auch die Überprüfung, ob mit Fehlerbedingungen korrekt umgegangen wird. Dabei werden Abhängigkeiten zu externen Ressourcen, wie beispielsweise einer Datenbank, außen vor gelassen. Wenn solche Ressourcen dennoch benötigt werden, müssen diese mit Hilfe eines sogenannten „Mocking-Framework“ simuliert werden. Schließlich stellt der CI-Server die Ergebnisse dieser Tests (z.B. in einem Jenkins-Dashboard) zur Verfügung. [\[Last 17\]](#)

„Continuous Delivery“

Bei Continuous Delivery werden die Änderungen, die vorher auf dem isolierten CI-Server getestet und integriert wurden, in eine Stage, z. B. in die Entwicklungsstage, automatisch übertragen. Hier werden weitere Tests, wie Integrationstests und Akzeptanztests, durchgeführt. Am Ende einer Continuous Delivery Pipeline steht ein theoretisch auslieferbarer Stand der Anwendung. Die Übergabe dieses Standes in die Produktion ist bei Continuous Delivery noch manuell umzusetzen. [\[Last 17\]](#)

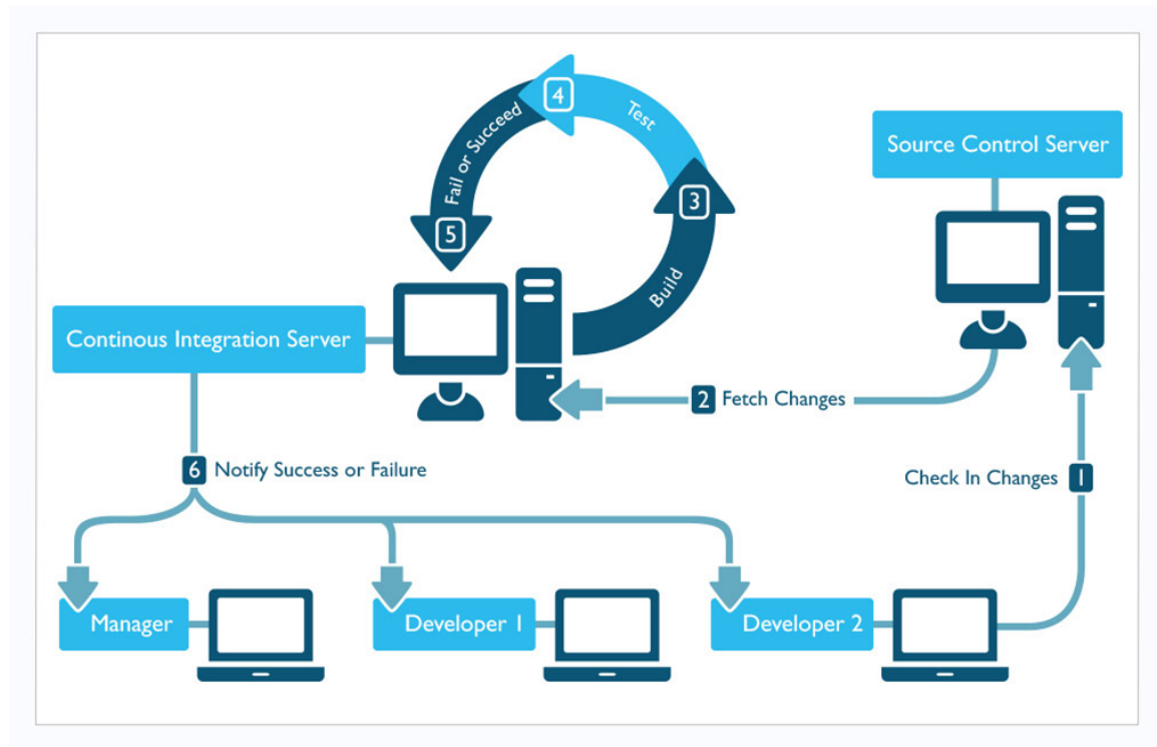


Abbildung 2.2.: Continuous Integration Prozessaufbau (Quelle: [<http> 20c])

„Continuous Deployment“

Continuous Deployment beschreibt den nächsten Schritt nach Continuous Delivery. Dabei handelt es sich um die automatisierte Bereitstellung in eine Produktions-Stage. [[Last 17](#)]

2.2. Mainframe / Großrechner

Im modernen Sprachgebrauch kann ein Großrechner oder auch Mainframe als größte zur Verfügung stehende Serverart betrachtet werden. Er wird von Unternehmen verwendet, um kommerzielle Datenbanken, Transaktionsserver und Anwendungen, die einen hohen Grad an Sicherheit und Verfügbarkeit benötigen, zu hosten. Im Gegensatz zu verteilten Serversystemen, bei denen die Funktionalitäten auf einzelne Server, wie zum Beispiel einen E-Mail-Server, einen Datenbank-Server, einen Web-Server usw. aufgeteilt sind, handelt es sich bei einem Mainframe um ein zentralisiertes System. Die einzelnen Funktionalitäten werden von sogenannte „Subsysteme“, auch „Middleware“ genannt, zur Verfügung gestellt. Darunter zählen unter anderem Datenbanksysteme und Anwendungsserver. [[Ebbe 11](#)]

2.3. Mainframe Anwendungen bei DATEV e.G.

Das Betriebssystem des IBM Mainframes ist das für zero downtime stehende z/OS.⁶ Darauf aufbauend benötigen klassische z/OS Anwendungen bestimmte Middleware. Bei der DATEV e.G. handelt es sich unter anderem um folgende Middlewarekomponenten:

- Laufzeitumgebung: CICS oder Batch
- Datenhaltung: VSAM oder Db2
- Message Queuing: IBM MQ

Diese Subsysteme stehen in jeder Stage zur Verfügung. Eine Stage beschreibt eine isolierte Systemumgebung mit eigenen Subsystemen und Ressourcenverwaltung. Die DATEV e.G. unterscheidet am Mainframe vier Stages:

- Testplex:
Labor für Änderungen am System, beispielsweise einer neuen Betriebssystemsversion
- Entwicklung:
Implementierung neuer Features und Durchführung kleiner Tests
- Qualitätssicherung:
Durchführung von Integrationstests
- Produktion:
Software, die für den Kunden bereitsteht

2.4. Subsysteme / Middleware

Für die Beantwortung der Forschungsfragen liegt der Fokus auf dem Erstellen („Provisionieren“) einer anwendungsspezifischen Laufzeitumgebung mit einer Datenhaltung und Message Queuing auf der Entwicklungs-Stage. Als Laufzeitumgebung wird „CICS“, als Datenhaltung „Db2“ und für das Message Queuing „IBM MQ“ verwendet. Wie in Abbildung 2.3 dargestellt ist, sind mehrere Instanzen pro Subsystem möglich. Bei der DATEV e.G. wird die Anzahl an Instanzen pro Subsystem möglichst gering gehalten, um den Verwaltungsaufwand und den Aufwand bei Änderungen, beispielsweise bei einem Versionswechsel⁷, gering zu halten. Daraus folgt, dass sich, wie in der Problemstellung, Absatz 1.1, bereits erwähnt, viele Anwendungen die gleichen Entwicklungs-CICS/Db2/IBM MQ Ressourcen teilen. Diese Instanzen sind langlebig, und müssen dahingehend gepflegt und gewartet werden, dass sie die Anforderungen für alle Anwendungen, die sich die Ressourcen teilen, abdecken.

⁶[Ebbe 11]

⁷circa alle eineinhalb Jahre erscheint eine neue CICS Version

Diese einzelnen Subsysteme werden im Folgenden erläutert, hierzu dient Abbildung 2.3 als Überblick.

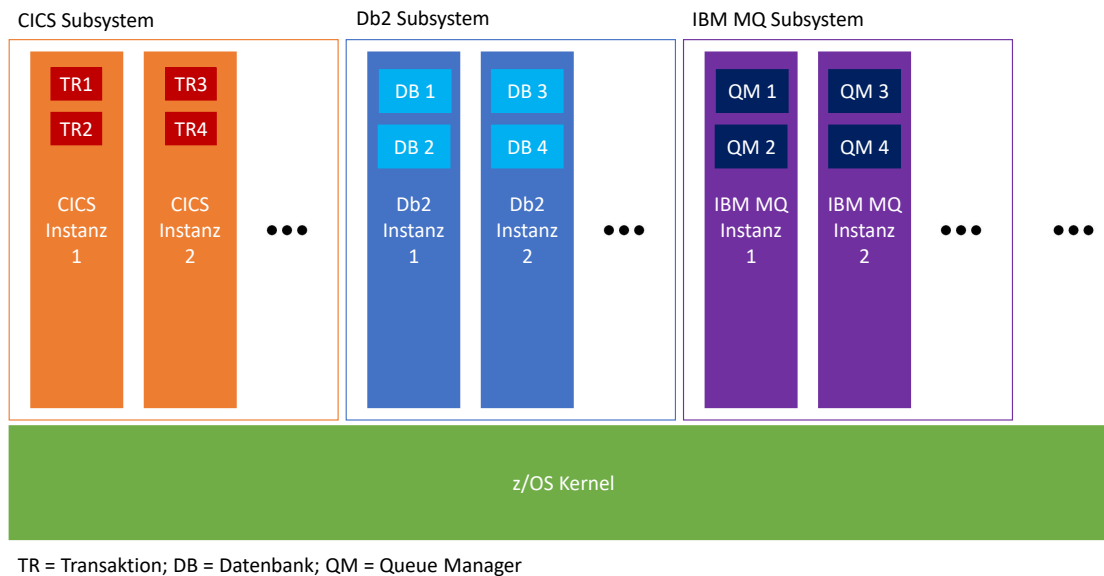


Abbildung 2.3.: Architekturübersicht über die Subsysteme einer Stage bei DATEV eG

2.4.1. Customer Information Control System

Das Customer Information Control System, kurz CICS, ist ein Applikationsserver für einen IBM-Großrechner mit Betriebssystem z/OS und damit eine IBM Middleware. Ein Applikationsserver stellt eine Umgebung zur Verfügung, in der Anwendungen gehostet werden können. Dabei kümmert sich dieser unter anderem um Transaktionalität, Webkommunikation und Sicherheit. Hierfür stellen Applikationsserver eine API zur Verfügung. CICS hat gegenüber anderen Anwendungsservern, wie zum Beispiel „Apache Tomcat“ oder „IBM WebSphere“, den Vorteil, dass es verschiedene Programmiersprachen unterstützt. Damit ist CICS ein Multi-Language Application Server und kann z.B. von COBOL, Assembler, Java und PLI Programmen genutzt werden. So können Programme innerhalb einer Anwendung in der für ihren Use-Case am besten geeigneten Sprache implementiert werden, für die Kommunikation zwischen den verschiedenen Sprachen stellt CICS mit seinem erwähnten API die Funktionalität zur Verfügung. [Rayn 11]

Das CICS Subsystem einer Stage umfasst mehrere CICS Instanzen.

2.4.1.1. CICS Instanz

Unter einer CICS Instanz ist ein einzelner Bereich, der auf dem z/OS Kernel aufsetzt, zu verstehen. Dieser Bereich ist mittels einer eindeutigen CICS ApplicationID gekennzeichnet und kann darüber explizit verwaltet werden. Eine CICS Instanz verwaltet mehrere CICS Transaktionen.

Wenn in dieser Arbeit von dem CICS gesprochen wird, ist damit die CICS-Instanz gemeint.

2.4.1.2. CICS Transaktion

Ein Businessablauf wird im CICS in einer Transaktion gekapselt. Eine Transaktion kann mehrere Programme unterschiedlicher Programmiersprachen umfassen und wird über eine eindeutige „TransaktionsID“ identifiziert..

Über die TransaktionsID wird der Ablauf gestartet. Dies kann sowohl per Webanfrage oder per Messaging Queue als auch aus einem anderen Programm heraus oder manuell geschehen. In der Transaktion werden alle Änderungen, die Programme an Ressourcen, wie zum Beispiel einer Datenbank oder Dateien tätigen, protokolliert. So wird im Falle eines Fehlers die Möglichkeit eines Rollbacks, beispielsweise der in der Transaktion genutzten Datenbank, sichergestellt. [[Rayn 11](#)]

2.4.2. Db2

Db2 ist ein relationales Datenbanksystem, welches unter anderem als Subsystem eines z/OS Betriebssystems läuft. Einer Stage können mehrere Datenbanksysteme, auch Instanzen genannt, zugeordnet werden. In einer Instanz befinden sich die Datenbanken und Tabellen. In der Entwicklungsstage sind das unter anderem „DB0C“ und „DB0T“. DB0C wird als Sandbox betrieben, d.h. jeder Entwickler kann hier eigene Datenbanken erstellen und verwaltet. DB0T enthält von der Administration für die Entwickler verwaltete Datenbanken. Diese Datenbanken entsprechen in der Struktur den Datenbanken aus der Produktion und werden für Tests bezüglich Strukturänderungen usw. herangezogen.

2.4.3. IBM MQ

IBM MQ ist eine Messaging-Lösung der IBM. Diese ermöglicht den asynchronen Datenaustausch zwischen Anwendungen mittels sogenannter Queues. Alle IBM MQ Begrifflichkeiten, die in dieser Arbeit verwendet werden, werden im Folgenden erläutert. [[Aran 13](#)]

Das IBM MQ Subsystem einer Stage setzt sich aus einem oder mehreren Queue Managern zusammen. Ein Queue Manager kann daher als IBM MQ Instanz gesehen werden.

2.4.3.1. Queue Manager

Bei einem Queue Manager handelt es sich um die zentrale Ressource eines IBM MQ Systems. Er verwaltet alle anderen IBM MQ Ressourcen. Dazu gehören unter anderem die Speichersteuerung der Daten und die Wiederherstellung dieser im Falle eines Fehlers. Desweiteren koordiniert er den Zugriff aller Anwendungen auf die Nachrichten in den von ihm verwalteten Queues. Um hierbei die Konsistenz sicherzustellen, sorgt er für Locking und die notwendige Isolation der Queues. [\[Aran 13\]](#)

2.4.3.2. Queues

In Queues werden die Nachrichten, die von Programmen gesendet und gelesen werden gespeichert. Es gibt verschiedene Arten von Queues, die im Kontext dieser Arbeit relevanten Queues sind folgende:

Die Local Queue.

Dabei handelt es sich um die einzige Queue Art, bei der die Nachrichten physikalisch gespeichert werden. Die anderen Queue Arten nutzen als Basis immer eine Local Queue.

Initiation Queue

Die sogenannte „Initiation Queue“ ist eine spezielle Art der Local Queue. Diese dient dem Queue Manager dazu, unter bestimmten Bedingungen eine Trigger-Nachricht darauf zu schreiben. Eine andere Local Queue kann so definiert sein, dass sobald eine Nachricht auf sie geschrieben wird, eine solche Trigger-Nachricht erzeugt wird. Dies ermöglicht beispielsweise den use-case, dass Anwendungen nur starten, wenn wirklich Daten zum Verarbeiten vorhanden sind. [\[Aran 13\]](#)

2.4.3.3. Process

Für das Auslösen von Anwendungen wird nicht nur die Initiation Queue benötigt, sondern auch sogenannte „Processes“. So muss der Local Queue, die den Start einer Anwendung auslösen soll, bei der Definition nicht nur die Initiation Queue bekannt gemacht werden, sondern auch ein Process. Ein Process legt den „Type“ und den Namen der zu startenden Anwendung fest. Als „Type“ können beispielhaft CICS oder auch WINDOWSNT für Windows unterstützte Plattformen genannt werden. Ist der „Type“ CICS, muss der Name der

Transaktion angegeben werden, für Windows Plattformen der Dateipfad der auszuführenden exe. [Aran 13]

2.5. „IBM Cloud Provisioning and Management for z/OS“

Die Verwaltung von Subsystemen von z/OS ist eine hochspezialisierte, teilweise manuelle Tätigkeit.⁸ Um diese Aufgaben zu bewältigen, sind aktuell in erster Linie proprietäre Tools (siehe Abbildung 2.4 im Einsatz, die nicht der Erwartungshaltung an moderne Administrationstools entsprechen.

```

OVERTYPE TO MODIFY                                CICS RELEASE = 0710
CEDA ALTER TRANsAction( MAIL )
  TRANsAction   : MAIL
  Group         : TESTPCT
  DEScription   ==> EMAIL MAILVERSAND TEST
  PROGram       ==> SPMail0
  TWAsize       ==> 00000          0-32767
  PROFile       ==> DFHCICST
  PArtitionset  ==>
  STATus        ==> Enabled        Enabled ! Disabled
  PRIMedsize    : 00000          0-65520
  TASKDATALoc   ==> Any            Below ! Any
  TASKDATAKey   ==> User            User ! Cics
  STOrageclear  ==> No              No ! Yes
  RUNaway       ==> System          System ! 0 ! 250-2700000
  SHutdown      ==> Enabled         Disabled ! Enabled
  ISolate       ==> Yes             Yes ! No
  Brexit        ==>
+ REMOTE ATTRIBUTES

                                SYSID=CI01 APPLID=TCICS01

PF 1 HELP 2 COM 3 END          6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL

```

Abbildung 2.4.: Etablierte Konfigurationsoberfläche am Beispiel bei Änderung einer Transaktion

Für die Automation dieser Prozesse bietet die IBM das „IBM Cloud Provisioning and Management for z/OS“ an. Dabei handelt es sich um ein bei der Installation von „z/OS Management Facility“⁹, kurz z/OSMF, mitgeliefertes Tool. z/OSMF ist wiederum Teil der Standardauslieferung des Betriebssystems z/OS. Es soll z/OS-Administration über moderne Oberflächen und APIs ermöglichen. z/OSMF (und damit auch „IBM Cloud Provisioning and Management for z/OS“) ist bei DATEV verfügbar, wird aber noch nicht flächendeckend eingesetzt. Die Administratoren kennen und vertrauen auf ihre herkömmlichen Tools, und sehen in der Umstellung immer auch Lernaufwand und Risiko. „IBM Cloud Provisioning

⁸Analyse des aktuellen Bereitstellungsprozesses, siehe in Absatz 4.2

⁹Siehe Absatz 2.5.1

and Management for z/OS“ ist auch außerhalb der DATEV e.G. kaum verbreitet. Deshalb besteht auch aus Sicht der IBM Interesse an Erfahrungsberichten zur Nutzung von „IBM Cloud Provisioning and Management for z/OS“. So wurde im Rahmen der neunzehnten „AMC“¹⁰ Tagung im IBM Client Center Böblingen am 16.01.2020 bei einem circa 30 minütigen Vortrag ein Arbeitsstand dieser Arbeit vorgestellt. Die Agenda der Veranstaltung ist im Anhang A.1 zu finden. Der AMC e.V. ist ein Förderverein für die akademische Ausbildung auf dem Mainframe.¹¹ Neben Vertretern der IBM nahmen an der Tagung Vertreter von Hochschulen und verschiedenen Unternehmen teil. Der Vortrag „kam ja sehr gut an und hat auch später noch zu Gesprächen geführt“¹². In den anschließenden Gesprächen wurde deutlich, dass sich neben der DATEV e.G. und der IBM auch andere Kundenfirmen für die automatisierte Provisionierung von z/OS Middleware mit „IBM Cloud Provisioning and Management for z/OS“ interessieren, aber noch kaum Erfahrung in dem Umfeld besteht.

Bei „IBM Cloud Provisioning and Management for z/OS“ stehen die sogenannten „Templates“ im Mittelpunkt. Mit Hilfe eines Templates können Instanzen erzeugt werden. Diese Instanz kann eine oder mehrere verschiedene Subsystem-Instanzen enthalten. Das Template selbst ist eine Art „Verwaltungseinheit“ und besteht aus drei Dateien:

„Manifest-File“

Im Manifest-File ist der Speicherpfad für das sogenannte „Aktiondefinitionsfile“ und das sogenannte „Variableinputfile“ angegeben. Erläuterungen zu diesen beiden Dateien folgen im Anschluss. Ein Template benötigt zur Provisionierung einen dazugehörigen Provisionierungsworkflow, der Speicherpfad von diesem Workflow wird in der Manifest-File angegeben. [[http 20m](#)]

Ein Workflow ist über eine XML Datei, das sogenannte „Workflowdefinitionsfile“, definiert. Diese lässt sich grob in zwei Teile untergliedern:

- Variablendefinition
- Steps

In der Variablendefinition werden, wie der Name schon sagt, alle Variablen, die für diesen Workflow notwendig sind definiert.

Ein Step beschreibt einen Teilablauf eines Workflows. Ein Workflow kann aus mehreren Steps bestehen. Die Steps werden in Definitionsreihenfolge ausgeführt. Allerdings können Bedingungen für die Durchführung eines Steps definiert werden. So ist es beispielsweise möglich, einen Step nur durchzuführen, wenn eine bestimmte Variable einen bestimmten Wert besitzt. Innerhalb eines Steps können sowohl interne und externe Scripte als auch JCLs

¹⁰Academic Mainframe Consortium e.V.

¹¹[[http 20a](#)]

¹²AMC Vorstand Ernst Lugert, 20.1.2020

und somit z/OS Programme ausgeführt werden. Darüber hinaus besteht die Möglichkeit REST-Calls auszuführen. Durch ein XML Schema wird sichergestellt, dass das Workflowdefinitionfile keine syntaktischen Fehler beinhaltet. Sowohl die Variablendefinition als auch die Steps können in externe XML Dateien ausgelagert werden. Dadurch können Variablen an einer Stelle im Template definiert und in alle Workflowdefinitionfiles aufgenommen werden. [\[Rott 18\]](#)

Zusätzlich kann in dem Manifest-File eine Beschreibung des Templates hinzugefügt werden. [\[http 20m\]](#)

„Aktiondefinitionfile“

Weitere optionale Aktionen, die ein Anwender mit einer Instanz eines Templates durchführen kann, werden in dem Aktiondefinitionfile festgelegt. Standardmäßig handelt es sich dabei um folgende Aktionen. Die Begriffe werden am Beispiel einer Template-Instanz, die eine Datenbank provisioniert, erläutert.

- `check_status`
Prüft den Status der Template-Instanz, beispielsweise ob die Datenbank erreichbar ist.
- `start`
Welche Schritte sollen beim Starten der Template-Instanz durchgeführt werden. Beispielsweise die Erzeugung von Tabellen.
- `stop`
Welche Schritte sollen beim Stoppen der Template-Instanz durchgeführt werden. Beispielsweise das Löschen von Tabellen.
- `deprovisioning`
Welche Schritte sollen beim Entfernen der Template-Instanz durchgeführt werden. Üblicherweise wird die Template-Instanz zunächst gestoppt, am Beispiel der Datenbank ist dies nicht notwendig, da beim Löschen der Datenbank automatisch die Tabellen auch gelöscht werden.

Neben den Workflowdefinitionfiles muss in einer Aktion auch der Pfad für das sogenannte „Variableinputfile“ angegeben sein. [\[http 20m\]](#)

„Variableinputfile“

In dieser Datei werden den in der Workflowdefinitionfile definierten Variablen Werte zugewiesen. Somit kann das Template für spezifische Anforderungen, z.B. einer speziellen Anwendung, konfiguriert werden.

Für die Provisionierung eines Templates müssen diesem eine sogenannte „Domain“ und ein „Tenant“ zugewiesen werden. Unter einer „Domain“ ist ein System zu verstehen, das Systemressourcen in Ressourcenpools gliedert. „Tenants“ sind die dazugehörigen Rechtegruppen, die dem Anwender den Zugriff auf und die Nutzung von zugeordneten Templates ermöglicht. [Rott 18]

„IBM Cloud Provisioning and Management for z/OS“ umfasst zwei Lösungen, „z/OS Management Facility“¹³ und „z/OS Provisioning Toolkit“¹⁴.

2.5.1. z/OS Management Facility

Der Funktionsumfang von z/OS Management Facility, kurz z/OSMF, umfasst Systemmanagementfunktionen in einer browserbasierenden Benutzeroberfläche, dargestellt in Abbildung 2.5. Zu diesen Funktionen zählt auch die Verwaltung von Workflows und Templates.

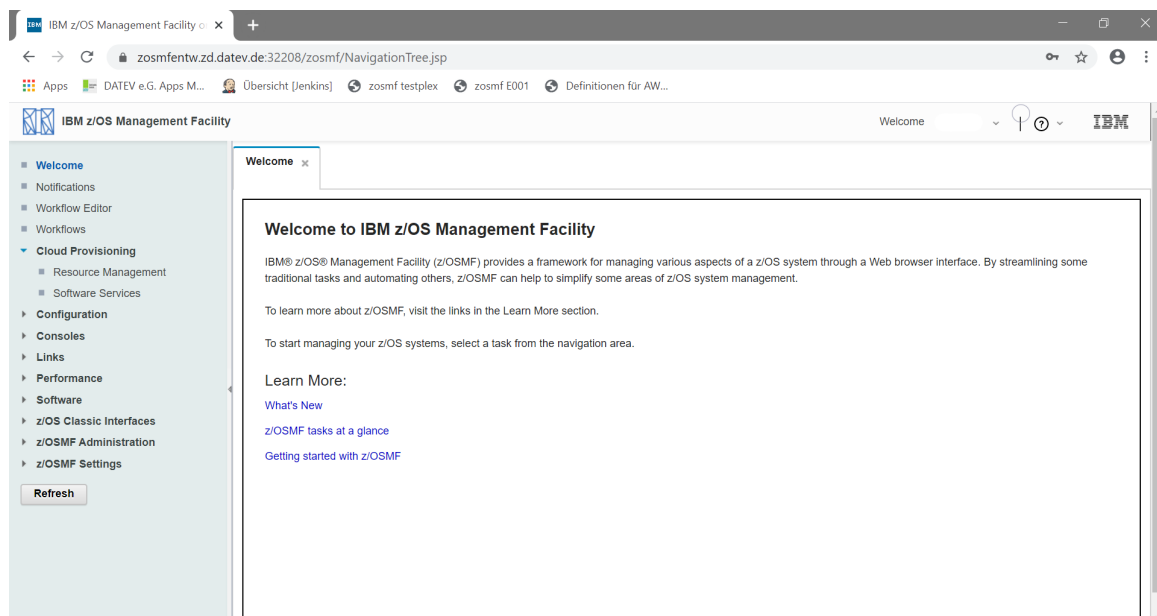


Abbildung 2.5.: z/OSMF Willkommens Ansicht

Die linke Seite der Abbildung 2.5 zeigt den Umfang der z/OSMF Funktionen. Für diese Arbeit besitzt nur der Menüpunkt „Cloud Provisioning“ Relevanz. Unter diesem Punkt sind die Funktionalitäten von „IBM Cloud Provisioning and Management for z/OS“ zu finden, und der Begriff „z/OSMF“ wird im folgenden synonym für diese Lösung verwendet [Rott 18]

Unter dem Punkt „Resource Management“ werden die „Domains“ und „Tenants“ verwaltet. Zur Verwaltung der Templates und Template-Instanzen kommen die „Software Services“

¹³siehe Absatz 2.5.1

¹⁴siehe Absatz 2.5.2

zum Einsatz. Dort können neue Templates über die Manifest-Files hinzugefügt werden. Es folgt, wie oben beschrieben, die Zuweisung einer „Domain“ und eines „Tenants“. Anschließend kann das Template, falls es keine Fehler beinhaltet, veröffentlicht werden. Es ist zu empfehlen vorher einen „Test Run“ durchzuführen. Dabei wird eine Instanz testweise provisioniert. Diese Test-Template-Instanz verhält sich genauso wie eine Instanz, die aus einem veröffentlichten Template erzeugt wurde. Somit können das Template und die in der Aktiondefinitionfile definierten Aktionen vor der Veröffentlichung getestet werden. [Rott 18]

2.5.2. z/OS Provisioning Toolkit

Das z/OS Provisioning Toolkit, kurz z/OSPT bietet für das Provisionieren von Laufzeitumgebungen ein Kommandozeileninterface für die Bereitstellung und das Verwalten von Templates bzw. „Images“ sowie das Starten von Instanzen an. z/OSPT orientiert sich an der Docker Commandline und spricht von Containern (auch wenn es in z/OS diese nicht gibt) und Images. Zur Erläuterung werden die Begriffe hier gegenüber gestellt.

„Docker-Images“

Ein Docker-Image beschreibt die Vorlage für einen Docker-Container und beinhaltet alle Elemente, die für die Ausführung einer Anwendung als Container benötigt werden, so wie den Code, Konfigurationsdateien, Umgebungsvariablen, Bibliotheken und die Laufzeitumgebung.

„Docker-Container“

Mit dem Kommandozeilenbefehl „docker run“ wird aus einem Docker-Image ein Docker-Container erzeugt. Ein Docker-Container beschreibt somit eine lauffähige Instanz eines Docker-Images. [Vohr 16]

Im Vergleich dazu sind die Definitionen in z/OSPT folgende:

„z/OSPT-Images“

Grundsätzlich ist ein z/OSPT-Image einem Docker-Image nicht unähnlich. Auch ein z/OSPT Image ist für die Erzeugung eines z/OSPT Containers zuständig und beinhaltet alle Elemente, die die Anwendung zur Laufzeit benötigt. Es verknüpft ein Template mit den enthaltenen Dateien (Actiondefinitionfile, Variableinputfile und die Manifest-File) mit einer weiteren nicht im Template enthaltenen Konfigurationsdatei, der sogenannten „zosptfile“. In dieser muss der Name des zugrundeliegenden Templates angegeben werden. Danach ist es möglich, die Werte aus der Variableinputfile zu überschreiben und so das Verhalten der Template-Instanz zu verändern. Dadurch kann ein Template mit spezifischen Änderungen provisioniert werden, ohne dass ein neues Template erzeugt werden muss.

„z/OSPT-Container“

Die Beziehung zwischen einem z/OSPT-Container und einem z/OSPT-Image ist die gleiche wie zwischen einem Docker-Container und einem Docker-Image. Ein z/OSPT-Container entspricht einer Template-Instanz, die mit Hilfe eines z/OSPT-Images gestartet wurde.

Um nun einen z/OSPT-Container bereitzustellen, muss ein Template zur Verfügung stehen. Anschließend kann mittels des Konsolenbefehls „zospt build“ und der Angabe des Pfades der zosptfile ein Image erzeugt werden. Wird nun der „zospt run“-Befehl ausgeführt, wird ein z/OSPT-Container erzeugt (entspricht dem Provisionieren einer Template-Instanz) und gestartet (nur wenn die start-Aktion in der Actiondefinitionfile definiert wurde). Der Status von vorhanden Instanzen und Container kann ebenfalls mittels Kommandozeilenbefehlen abgefragt werden. [?] In Abbildung 2.6 werden die möglichen Kommandozeilenbefehle mittels des Befehls „zospt -h“ in einem Kommandofenster angezeigt.

```
$ zospt -h
IBM z/OS Provisioning Toolkit V1.1.5

Usage: zospt [OPTIONS] COMMAND [arg...]

Options:
  --version      : Displays the command line version.
  -h (--help)    : Displays the command line help.

Commands:
  build          PATH [-h (--help)] -t (--tag) <imageName>          Build an image
  images         [-h (--help)]                                     List all images
  inspect        <imageName> | <containerName> | <containerId>      Inspect an image or a container
  rm             <containerName> | <containerId> ... [-f (--force)]  Remove one or more containers
  rmi            <imageName> ... [-h (--help)]                     Remove one or more images
  run            <imageName> [--draft]                             Run an image in a new container
  start          [--link <containerName> | <containerId>:<alias>]
                  [--name <containerName>] [-h (--help)] [-q (--quiet)]
  stop           <containerName> | <containerId> ... [-h (--help)]  Stop one or more containers
  ps            [-a (--all)] [-f (--filter) <filter>] [-h (--help)] List containers

Run 'zospt COMMAND --help' for more information on a command.
```

Abbildung 2.6.: z/OSPT mögliche Kommandozeilenbefehle

Kapitel 3.

Vorgehensweise

Für die Beantwortung der Forschungsfrage „Ist es möglich, den Bereitstellungsprozess für z/OS Anwendung bei DATEV e.G. mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“-Tools an cloud native Prozesse anzunähern?“ werden zunächst die Vorteile des cloud native Prozesses bei der DATEV e.G. dargestellt. Um letztendlich eine mögliche Verbesserung des momentan in der DATEV e.G. etablierten Bereitstellungsprozesses für die Middlewarekomponenten CICS, Db2 und IBM MQ bewerten zu können, wird dieser analysiert.

Wie in Absatz 1.2 beschrieben, wird für die Beantwortung der Forschungsfragen eine Beispielanwendung, die als Laufzeitumgebung CICS, als Datenbank Db2 und als Message Lösung IBM MQ verwendet, benötigt. Hier bietet sich die „DATEV Rechnungsschreibung“¹ an. Dabei handelt es sich um eine legacy z/OS Anwendung, ein Teil dieser Anwendung erfüllt die oben genannten Kriterien. Um die genauen Anforderungen an die Middleware zu kennen, wurde dieser Teil analysiert. Anhand dieser Anforderungen soll mittels z/OSPT oder z/OSMF eine individuelle, auf die Anwendung zugeschnittene Laufzeitumgebung bereitgestellt werden. Konkret verlangt dies die Implementierung eines Templates. Für die Entscheidung, ob z/OSPT oder z/OSMF zum Einsatz kommt, wurden die Vor- und Nachteile beider Lösungen gegenübergestellt.

Im Folgenden wird speziell auf die Vorgehensweise bei der Implementierung des Templates eingegangen.

Um die von allen Entwicklern bei DATEV e.G. verwendeten Subsysteme der Entwicklungsstage bei eventuell auftretenden Fehler in der neuen, automatisierten Bereitstellung nicht zu beeinflussen, wurden die ersten Schritte auf dem Testplex durchgeführt. Die Anwendung selbst kann auf dem Testplex nicht verprobt werden, da die notwendigen Testdaten auf Db2 dort nicht verfügbar sind. Deshalb wird dort vorerst nur die Provisionierung der benötigten Middleware untersucht. Damit wird das Ziel verfolgt, erste Erfahrungen mit „IBM Cloud Provisioning and Management for z/OS“ zu sammeln und ein Template zu provisionieren, das DATEV spezifische Middleware, also CICS, Db2 und IBM MQ Instanzen beinhaltet.

¹Beschreibung siehe Absatz 4.3

Da das CICS Subsystem als Laufzeitumgebung im Mittelpunkt der Subsysteme steht, werden zunächst folgende, von der IBM bei der Installation von z/OSMF mitgelieferten, CICS Templates untersucht:

- „cics_getting_started“
- „cics_54“

„cics_getting_started“

Dieses Template wird von der IBM für die ersten Schritte der Provisionierung einer CICS Instanz angeboten. Das Template bietet nur minimale Konfigurationsmöglichkeiten. So entspricht die CICS Instanz einer minimal lauffähigen CICS Instanz nach IBM Standard.

„cics_54“

Hierbei handelt es sich um ein Template, das eine vollumfängliche CICS Instanz nach IBM Standard mit der CICS Version 5.4 provisioniert. Es ermöglicht die Angabe von komplexen Konfigurationen.

Mit den durch die Untersuchung dieser beiden Templates gesammelten Erfahrungen erfolgt die Implementierung eines an das DATEV e.G. Umfeld angepassten CICS Templates. Ist die daraus erzeugte CICS Instanz funktionsfähig, wird die Provisionierung einer Db2 Datenbank und IBM MQ Queues nacheinander in das Template aufgenommen. Für ein Erzeugen von Db2 und IBM MQ existieren bei DATEV e.G. bereits einzelne, voneinander unabhängige Services der jeweiligen Administrations-Teams, die als Bausteine in das Template integriert werden können. Für die Provisionierung von Db2 Datenbanken existiert z.B. bereits eine REST-API.

IBM MQ Queues werden mittels Standard IBM Jobs provisioniert.

Nachdem eine CICS Instanz, eine Db2 Datenbank und IBM MQ Queues auf dem Testplex sowohl provisioniert als auch deprovisioniert werden können, folgt der nächste Schritt. Dabei handelt es sich um den Wechsel vom Testplex in die Entwicklungsstage. In der Entwicklungsstage sind alle Anwendungsdaten, die für Anwendungstests notwendig sind, vorhanden. Somit kann hier die Integration der Beispielanwendung in die provisionierte Laufzeitumgebung sowie ein Testlauf stattfinden.

Um ein Meinungsbild bezüglich des Einsatzes von „IBM Cloud Provisioning and Management for z/OS“ bei DATEV e.G. zu bekommen, wurden mit Stakeholdern, also Mitarbeitern der Administratorenteams, Entwicklern und dem Technologiestrategieteam semi-strukturelle Interviews durchgeführt. Aus den Administratorenteams und von den Entwicklern wurden jeweils zwei Vertreter und aus dem Technologiestrategieteam ein Vertreter befragt. Es handelt sich hier um Experten in ihrem Arbeitsbereich, es ist zu beachten, dass

hier generell nur eine geringe Anzahl von Kollegen für Befragungen zur Verfügung steht. Es wurden semi-strukturelle Interviews gewählt um auf einzelne Antworten genauer eingehen zu können. Auf eine Transkription der Interviews wurde verzichtet, da nicht der genaue Verlauf der Interviews, sondern die Antworten auf die Fragen relevant sind.

Die Interviews fanden in Besprechungsräumen innerhalb eines DATEV e.G. Gebäudes statt und dauerten etwa dreißig bis vierzig Minuten. Vor den eigentlichen Interviews wurde sowohl die z/OSMF Lösung (siehe Absatz 5.4) als auch die durch z/OSPT ermöglichte Lösung (siehe Absatz 6) den kompletten Teams erläutert. Der Schwerpunkt der Vorstellung wurde an die jeweilige Zielgruppe (Entwickler, Administrator, Technologie-Strategie) angepasst. So wurde bei den Administratorenteams vor allem auf die Erstellung der Templates, welche für ihr Arbeitsgebiet relevant ist, eingegangen. Sowohl der Fragenkatalog, als auch die ausgefüllten und digitalisierten Fragebögen sind im Anhang A.4 zu finden. Für das Entwicklerteam und der Mitarbeiterin des Technologiestrategieteams waren nur die Fragen 1., 2. und 6. bis 10. des Fragebogens von Relevanz.

Kapitel 4.

Analyse

Im Folgendem wird der Einsatz von cloud native bei der DATEV e.G. analysiert. Im Vergleich dazu, wird der aktuelle Bereitstellungsprozess für die Laufzeitumgebung, dem dazugehörigen Datenbanksystem und einer Messaging Lösung erläutert. Anschließend erfolgt eine Beschreibung der Beispielanwendung „DATEV-Rechnungsschreibung“. Die dafür benötigten Informationen stammen aus Gesprächen mit Mitarbeiter 1 aus der Abteilung, die für die DATEV-Rechnungsschreibung zuständig ist. Es wird vor allem der technische Aspekt beleuchtet.

4.1. Analyse von cloud native bei DATEV e.G.

Neben den im Absatz 2.1 beschriebenen Begriffen, „cloud native“, „Cloud Foundry“ und „CI/CD“ ist der sogenannte „DevOps“-Gedanke in diesem Zusammenhang bei der DATEV e.G. von Bedeutung. Laut dem Buch „The Phoenix Project“ von Gene Kim¹ lassen sich die Werte und die Philosophie von DevOps in „Three Ways“ erläutern.

„The First Way“ betrachtet den Fluss von der Entwicklung über die Middleware Verwaltung bis hin zum Kunden. Um diesen Fluss zu optimieren, sollten möglichst kleine Änderungen in möglichst kleinen Arbeitsabschnitten entwickelt werden. Dabei helfen unter anderem Continuous Integration², Continuous Delivery³ und das automatische Erstellen von Laufzeitumgebungen nach Bedarf.

„The Second Way“ betrachtet den Feedback-Fluss über alle Stages hinweg. Ziel dabei ist, möglichst schnell Fehler zu erkennen und diese zu beheben oder gar zu vermeiden. Zur Erreichung dieses Ziels gehört unter anderem das Abschaffen der sog. „Schubladendenkweise“. Ein Beispiel aus Entwicklersicht:

¹[Kim 14]

²Siehe Absatz 2.1.2

³Siehe Absatz 2.1.2

„Meine Anwendung läuft nicht... Ach da gibt es Probleme mit dem Webserver, dass werden die Kollegen der Administration schon lösen“.

„The Third Way“ betrachtet weniger den Prozess an sich, sondern eher das Schaffen einer Kultur. Eine Kultur, die erlaubt Risiken einzugehen und aus Fehlern zu lernen, aber auch das Verständnis das Wiederholung und Übung Voraussetzung für einen guten Entwickler sind. Dabei ist ein hoher Grad an Vertrauen in das Team notwendig.

Bei der DATEV e.G. wird dieser DevOps-Gedanke durch sogenannte „Product-Teams“ umgesetzt. In einem solchen Team kümmert man sich neben der eigentlichen Softwareentwicklung, Testmaßnahmen usw. auch um den Betrieb der Middleware, z.B. die im Cloud Native Umfeld üblichen noSQL Datenbanken wie Mongo oder PostgreSQL. Dem ganzen Team wird viel Vertrauen gegeben und es soll weitestgehend autonom Software erstellen und betreiben können. Dadurch steigt die Effektivität des Teams.

Die Kombination mit einer automatisierten CI/CD-Pipeline birgt weitere Vorteile. So sorgt Continuous Integration dafür, dass individuelle Codeänderungen sofort nach der Integration in die Codebasis auf Korrektheit geprüft werden. Im Fehlerfall werden die Entwickler sofort automatisch benachrichtigt. Verbunden mit regelmäßiger Integration führt das zu einer schnelleren und einfacheren Fehlerbehebung. Ein weiterer Vorteil ist, dass neben der Prüfung auf Korrektheit auch eine statische Codeanalyse möglich ist. Dadurch kann sichergestellt werden, dass der Code den gewünschten Qualitätskriterien entspricht. Durch Continuous Delivery und Continuous Deployment kommen diese Änderungen im Idealfall voll automatisiert beim Kunden an. Auch dies führt zu einer Effizienzsteigerung.

Für die Bereitstellung der benötigten Laufzeitumgebung kommt bei DATEV e.G. die Cloud Foundry Distribution von „pivotal“ als PaaS Plattform zum Einsatz. Durch den darin zur Verfügung gestellten „Marketplace“ können Laufzeitumgebungen effizient „auf Knopfdruck“ generiert werden. Die automatisierte Bereitstellung ist mittels des Kommandozeileninterfaces und der Integration in eine CI/CD-Pipeline möglich. So werden z.B. für jeden Entwickler isolierte Testumgebungen geschaffen.

4.2. Aktueller Bereitstellungsprozess

Im Vergleich zu den Product-Teams im cloud native Umfeld bei denen sich das Team selbst um den Betrieb der Middleware kümmert, verwalten im z/OS Umfeld eigen dafür entstandene Administratorenteams die Middleware - stage-übergreifend. Die „CICS Administration“ kümmert sich um alles rund um das CICS Subsystem. Die „Db2 Administration“ stellt Datenbanken und Tabellen auf Anfrage der Entwickler bereit. Die „IBM MQ Administration“

verwaltet die IBM MQ Ressourcen. Um die Stellen, an denen das „IBM Cloud Provisioning and Management for z/OS“-Tool helfen kann, ausfindig machen zu können, wird im Folgenden der aktuell etablierte Prozess für z/OS Anwendungen beschrieben. Die in diesem Absatz genannten Informationen stammen aus Gesprächen mit Mitarbeitern aus den jeweiligen Administratorenteams. Wie in Absatz 2.3 beschrieben benötigt eine z/OS Anwendung zunächst eine Laufzeitumgebung, im Fall dieser Arbeit handelt es sich um CICS.

4.2.1. Bereitstellung einer CICS Instanz

Um eine lauffähige CICS-Instanz einzurichten, sind mehrere Schritte notwendig. Der komplette Prozess wird in Abbildung 4.1 dargestellt. Wie zu sehen ist, ist der Initiator des Prozesses das Entwicklerteam. Zunächst wird dort während der Entwicklungsphase festgestellt, dass eine neue CICS-Instanz benötigt wird. Hier hilft das CICS Administratorenteam mittels Beratung aus. Während einer Beratungsphase, die via Telefon, Emails oder Terminen stattfindet, wird sichergestellt, ob wirklich eine neue CICS-Instanz notwendig ist oder ob nicht eine bereits bestehende Instanz genutzt werden kann. Falls eine neue CICS-Instanz benötigt wird, wird ein RACF Eintrag für diese Instanz beantragt. Dieser Eintrag wird dann vom RACF Team erzeugt. Um sicherzustellen, dass die CICS-Instanz in den täglichen Sicherungen enthalten ist, muss das System Automations-Team benachrichtigt werden.

Nun kann mit dem eigentlichen Anlegen der CICS-Instanz begonnen werden. Dabei müssen folgende Schritte manuell durchgeführt werden. Es werden nur die Schritte, die im Laufe dieser Arbeit durch das „IBM Cloud Provisioning and Management for z/OS“-Toolkit automatisiert werden, dargestellt. Es handelt sich um das Anlegen von:

- CICS spezifische Dateien
- „CICS System Definition“
- Started Task Control-Job

CICS spezifische Dateien

Zunächst müssen CICS spezifische Dateien im z/OS angelegt werden. Im Fall dieser Arbeit zugrunde liegenden Beispiels handelt es sich um siebzehn verschiedene VSAM⁴ Dateien. Diese Dateien benötigt die CICS-Instanz um zum Beispiel Systemfehler zu protokollieren oder den Debugger aktivieren zu können.

⁴Virtual Storage Access Method, spezielle Dateiarart, die schnelle I/O-Zugriffe ermöglicht.[[Love 13](#)]

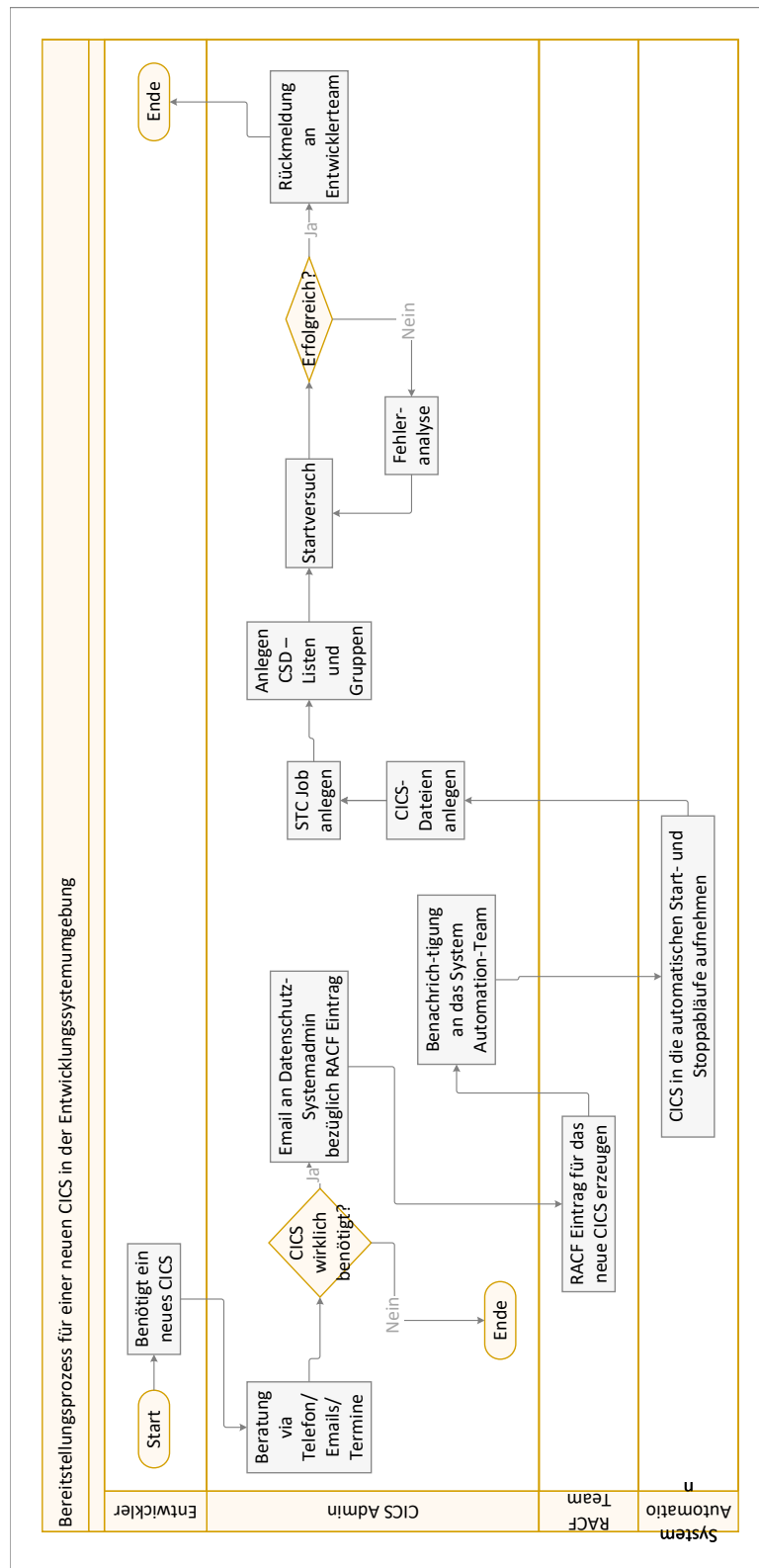


Abbildung 4.1.: Bereitstellungsprozess einer CICS Instanz

„CICS System Definition“

In der Datei „CICS System Definition“, kurz CSD, muss jede Ressource, die dem System zur Verfügung stehen soll, definiert werden. Eine CSD Datei kann für mehrere CICS-Instanzen verwendet werden und besteht aus mehreren Einträgen. Ein Eintrag besteht aus einer Gruppe und einer Liste. Die Gruppe ist hierbei die Definition einer Systemressource und muss manuell angelegt werden. Bei der Liste handelt es sich um das System, welches diese Ressource benötigt. Dort ist unter anderem für jede CICS-Instanz hinterlegt, zu welchem Db2 Datenbanksystem und welchem IBM MQ Messagingsystem sich diese Instanz verbinden soll.

Started Task Control-Job

Bei einem Started Task Control-Job, kurz STC Job, handelt es sich um einen Batch Job, der mit Hilfe des „START“-Konsolenkommandos innerhalb von z/OS gestartet werden kann. Dieser Batch Job wird deshalb auch als Started Task bezeichnet.[\[Cass 07\]](#) Bei der DATEV e.G. existiert für jede Instanz eines Subsystems ein solcher Job, so also auch für CICS. In diesem werden zunächst einige zur Laufzeit benötigten Bibliotheken und Dateien eingebunden, unter anderem die CICS spezifischen Dateien⁵. Außerdem werden hier die SIT⁶ Parameter definiert. Zunächst wird festgelegt welche Standard SIT verwendet werden soll. Anschließend können diese Standardwerte überschrieben werden. Zu diesen Parametern zählen unter anderem der eindeutige Name der CICS-Instanz, der Speicherort der dazugehörigen CSD und die Information, ob eine Verbindung zu einem Db2 Datenbanksystem hergestellt werden soll.

Nach der Durchführung dieser Schritte und einem erfolgreichen Startversuch, steht dem Entwickler eine neue CICS-Instanz zur Verfügung. Der komplette Ablauf dauert, unter der Annahme, dass alle Beteiligten verfügbar sind, nur diese Anforderung umsetzen müssen und für die Beratung ein Arbeitstag veranschlagt wird, circa zwei Arbeitstage.

4.2.2. Bereitstellungsprozess einer Db2 Datenbank

Wie in Abbildung 4.2 zu erkennen ist, ist der Bereitstellungsprozess einer neuen Db2 Datenbank mit vielen Aufgaben im Entwicklerteam verbunden. Zunächst müssen sogenannte Projektinformationen, unter anderem Daten der Voruntersuchung, vom Entwicklerteam bereitgestellt werden. Das Projektkürzel, der Datenbank- und Projektname und die Projektbezeichnung müssen mit den involvierten Abteilungen besprochen werden. Über den sogenannten „Datenbankänderungsantrag“ wird ein Genehmigungsprozess angestoßen. Wenn alle Genehmigungen erteilt wurden, kann ein Dateneigentümer festgelegt werden. Anschließend muss die Datenbank mittels eines Datenbankmodells vom Entwicklerteam beschrieben

⁵Beschreibung in Absatz 4.2.1

⁶CICS system initialization table

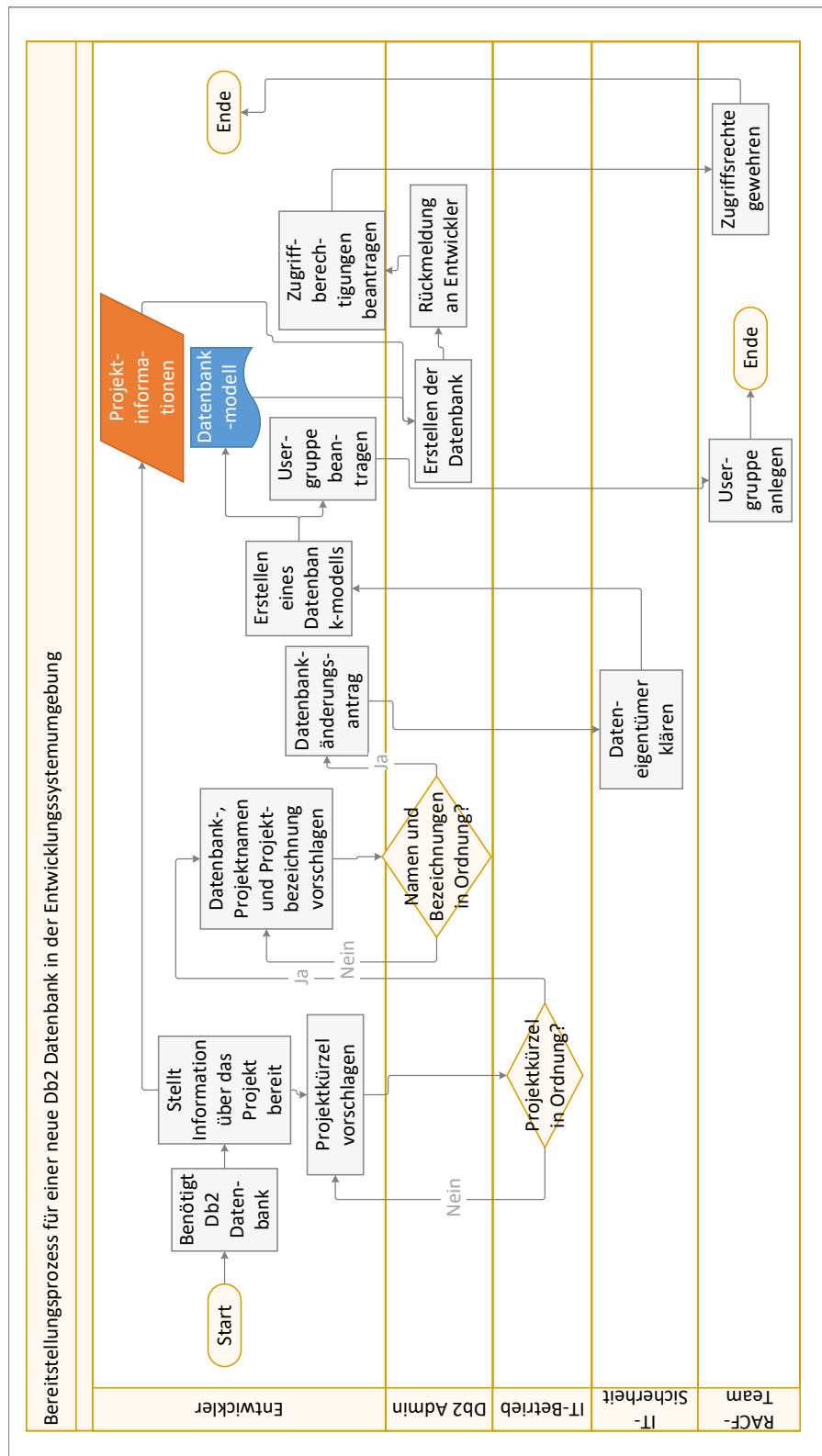


Abbildung 4.2.: Bereitstellungsprozess einer Db2 Datenbank

werden und eine Usergruppe, die im späteren Verlauf die Datenbankzugriffsrechte benötigt, beantragt und angelegt werden. Die eigentliche manuelle Erstellung der Datenbank wird mittels des Datenbankmodells und den Projektinformationen im Anschluss dazu durchgeführt.

Die Zugriffsrechte für die zuvor beantragte Usergruppe auf die neue Datenbank werden beantragt. Schließlich steht dem Entwicklerteam die neue Db2 Datenbank zur Verfügung. Wird die Db2 Datenbank in Verbindung mit einem CICS verwendet, so sind weitere manuelle Schritte vom CICS Administratorenteam notwendig. Der komplette Ablauf dauert, unter der Annahme, dass alle Beteiligten verfügbar sind, nur diese Anforderung umsetzen müssen und für die Beratung ein Arbeitstag veranschlagt wird, circa zwei Arbeitstage.

4.2.3. Bereitstellungsprozess einer IBM MQ Queue

Auch bei dem Bereitstellungsprozess, siehe Abbildung 4.3, einer IBM MQ Queue ist das Entwicklerteam der Initiator.

Die Grundlage dieses Prozesses ist ein Antrag auf Erstellung einer neuen IBM MQ Queue. Zuvor findet eine Beratung via Telefon, Email oder Terminen statt. Die Queues werden anschließend manuell eingerichtet und stehen dem Entwicklerteam zur Verfügung. Wird die Queue in Verbindung mit einem CICS verwendet, so sind weitere manuelle Schritte vom CICS Administratorenteam notwendig. Trotz des scheinbar schmalen Prozesses dauert der Ablauf unter der Annahme, dass alle Beteiligten verfügbar sind, nur diese Anforderung umsetzen müssen und für die Beratung ein Arbeitstag veranschlagt wird, circa zwei Arbeitstage.

4.2.4. Zusammenfassung aktueller Bereitstellungsprozess

Wie in den drei Diagrammen, Abbildungen 4.1, 4.2 und 4.3, zu erkennen ist, ist der aktuelle Bereitstellungsprozess noch mit vielen manuellen Schritten verbunden. Außerdem ist der Hauptaufwand in den Administratorenteams angesiedelt. Das Entwicklerteam ist der Initiator des Ablaufs. Folglich kümmert es sich um Formulare und die erste Kontaktaufnahme zum Administratorenteam.

Zusätzlich zu den vielen manuellen Schritten sind die vielen Absprachen zwischen mehreren Abteilungen zu nennen. Steht ein beteiligtes Team nicht zu Verfügung, kommt es zu Verzögerungen, das Team muss warten, der komplette Zeitplan kann sich dadurch nach hinten verschieben. Der Prozess für die Bereitstellung einer CICS-Instanz, mit einer Db2 Datenbank und IBM MQ Queues dauert in der Summe circa sechs Arbeitstage. Es setzt sich aus der Dauer der Einzelprozesse zusammen, für jedes Subsystem wird mit circa zwei Arbeitstagen gerechnet. Natürlich ist ein parallelisierter Ablauf der einzelnen Teilprozesse

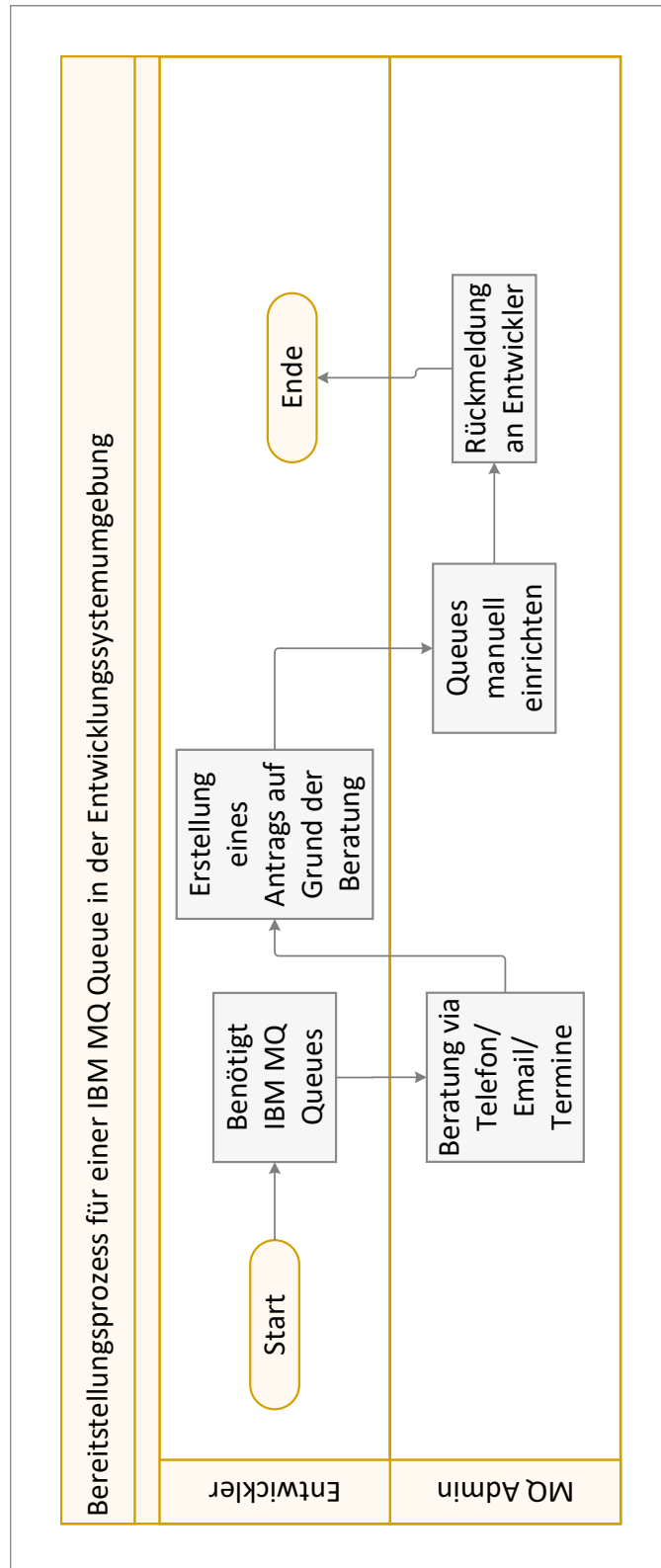


Abbildung 4.3.: Bereitstellungsprozess einer IBM MQ Queue

möglich, so kann die Gesamtdauer im besten Fall auf circa zwei bis drei Arbeitstage verkürzt werden.

Ein weiterer Punkt ist, dass die Kommunikation beziehungsweise der Initiator für den Start des gesamten Prozesses meist per Zuruf stattfindet. So existiert für die erste Kontaktaufnahme kein Formular, keine Automation oder ähnliches. Zur Kommunikation wird auf E-Mail, Telefon oder mittels Terminen zurückgegriffen.

Wird der Vergleich zu dem Bereitstellungsprozess im cloud native-Umfeld⁷ gezogen, so steht dem automatisierten und dadurch effizienten cloud native Prozess, ein durch viele Absprachen langsamer und fehleranfälliger Prozess gegenüber. Es ist festzustellen, dass die DATEV e.G. im Mainframe Umfeld noch weit vom DevOps-Gedanken entfernt. Product-Teams sind nicht vorhanden, die Administratorenteams sind für den Betrieb der Subsysteme verantwortlich und werden es in naher Zukunft auch weiterhin sein.

Neben diesen Problemen im Bereitstellungsprozess ist die grundlegende Architektur einer z/OS Anwendung zu betrachten. Im Vergleich zur „Microservices“-Architektur, die im cloud native Umfeld zum Einsatz kommt⁸, sind klassische z/OS Anwendungen monolithisch aufgebaut. Außerdem haben sich über die Jahre viele Abhängigkeiten und Verzahnungen zwischen einzelnen Anwendungen gebildet. Diese hohe Koppelung zwischen den Modulen erhöht die Komplexität und erschwert das Testen dieser Anwendungen. Um das Testen effizienter zu gestalten und einen Schritt in Richtung DevOps, einer CI/CD-Pipeline und dadurch einer Verkürzung der Releasezyklen auch für monolithische z/OS Anwendungen zu gehen, wird mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“-Tools der Aspekt der automatisierten Bereitstellung von Laufzeitumgebungen für die Entwicklungsphase untersucht. Als Beispielanwendung dient ein Teil der z/OS Anwendung „DATEV-Rechnungsschreibung“.

4.3. DATEV-Rechnungsschreibung

Für diese Arbeit wurde die DATEV-Rechnungsschreibung als Beispielanwendung herangezogen, weil sie folgenden Anforderungen entspricht. Es handelt sich zum einen um eine in sich abgeschlossene Anwendung, die nur zu Beginn des Prozesses von anderen Anwendungen abhängig ist. Zum anderen benötigt die DATEV-Rechnungsschreibung ein CICS als Laufzeitumgebung, eine Db2-Datenbank und IBM MQ als Messaginglösung. Somit kann ein umfangreicher Bereitstellungsmechanismus untersucht werden.

Bei dem Gesamtablauf handelt es sich um einen Batch-Ablauf auf dem Großrechner der DATEV e.G. Dieser setzt sich aus folgenden Teilen zusammen:

⁷Siehe Absatz 4.1

⁸Siehe Absatz 2.1

- Sammeln von Berechnungssätze
- Tägliche Bewertung
- Rechnungsaufbereitung

Für die Beantwortung der Forschungsfragen ist nur ein Teil der „Tägliche Bewertung“ relevant, die Preisermittlung.

4.3.1. Tägliche Bewertung

Dieser Ablauf läuft einmal täglich von Montag bis Freitag und ist für die Preisermittlung und Kundenzuordnung zuständig. Zur Realisierung wurden die Programmiersprachen Assembler, COBOL und Java genutzt. Am Ende dieses Ablaufes steht die ARUBA⁹-Db2-Datenbank. Dort werden die Berechnungsdaten der letzten 36 Monate aufbewahrt. Dabei handelt es sich um insgesamt circa 3,8 Milliarden Datensätze von einer Gesamtgröße von circa 400 GB mit Indizes. Diese Datensätze beinhalten alle Informationen für die endgültige Erzeugung der Rechnungen.

4.3.2. Preisermittlung

Die Preisermittlung ist für die Berechnung der Preise mit den dazugehörigen kundenindividuellen Abhängigkeiten, beispielsweise Rabatte, zuständig. Die Eingabe beläuft sich an Lasttagen auf bis zu 180.000 Geschäftspartner. Im DATEV e.G. Umfeld ist ein Geschäftspartner entweder eine Kanzlei oder ein einzelner Mandant. Aufgrund dieser Last wurde die Berechnung zum einen in CICS-Instanzen ausgelagert und zum anderen wurde der Ablauf in zwei Teile zerlegt:

- Bereitstellen der Preisinformationen
- Berechnung der Preise

Bereitstellen der Preisinformationen

Bevor die eigentliche Ermittlung der Preise stattfindet, werden zunächst die Preisinformationen und die kundenindividuellen Preisabhängigkeiten, wie zum Beispiel Rabatte, ermittelt. Für die Verarbeitung werden zwei Queues verwendet. Eine startet eine Transaktion im CICS, die andere wartet auf deren Antwort. Innerhalb der Transaktion werden alle benötigten Preisinformationen und -abhängigkeiten mit Hilfe einer Db2 Datenbank ermittelt. Diese Informationen werden dann in einem sogenannten „SHARED GETMAIN“-Bereich gespeichert. Dabei handelt es sich im Prinzip um einen Hauptspeicherbereich, der dem

⁹Abrechnungs- und Umsatz-Basis

CICS Subsystem zur Verfügung steht. Die Adresse dieses Bereiches wird den Transaktionen zur Verfügung gestellt. Somit greifen die einzelnen Transaktionen nicht mehr direkt auf die Datenbank zu, sondern stattdessen auf den schnelleren Hauptspeicher. Diese Vorarbeit ist notwendig, da es aufgrund von bis zu 60 Millionen Datenbankzugriffen zu massiven Einbußen bezüglich der Performance führen würde.

Berechnung der Preise

Um die Berechnungsdaten der 180.000 Geschäftspartner an CICS-Instanzen zu übertragen, stehen dem System weitere Queues zur Verfügung. Darunter ist eine allgemeine Queue in der alle Aufträge, die für die Weiterverarbeitung zur Verfügung stehen, geschrieben werden. Pro Geschäftspartner wird ein Auftrag angelegt. In diesem Auftrag befinden sich die Namen vier weiterer Queues. Eine dieser Queues beinhaltet alle Informationen, die für die Preisermittlung des dazugehörigen Geschäftspartners notwendig sind. Hierzu zählt unter anderem die Adresse des vorher beschriebenen Hauptspeicherbereichs. In den restlichen drei Queues sind die Ergebnisse der Preisermittlung gespeichert. Die Ergebnisse stehen somit dem Batch-Ablauf zur Weiterverarbeitung zur Verfügung. Für jede der vier Queues existieren jeweils 100 vorgefertigte Namen. Somit können auch maximal nur 100 Aufträge gleichzeitig auf Weiterverarbeitung warten. Falls dieses Limit erreicht ist, wartet der Batch-Ablauf so lange, bis einer der Aufträge fertig gestellt wird. Sobald ein Auftrag in die allgemeine Auftragsqueue geschrieben wird, wird eine CICS-Transaktion gestartet. Diese führt die Preisermittlung durch und schreibt das Ergebnis auf die dazugehörigen Queues. Ist dies geschehen, stehen die Queues wieder für einen neuen Auftrag zur Verfügung. Es können maximal 30 Transaktionen zeitgleich arbeiten.

Kapitel 5.

Realisierung

In diesem Kapitel wird die Implementierung eines Templates zur Beantwortung der Forschungsfragen der Arbeit¹ beschrieben. Dazu wird nach der im Kapitel 3 beschriebenen Reihenfolge der Arbeitsschritte vorgegangen. Es ist noch einmal zu erwähnen, dass zunächst abzuwägen ist, welche Implementierungsvariante des „IBM Cloud Provisioning and Management for z/OS“ besser geeignet ist: z/OSPT oder z/OSMF. Die Provisionierung einer CICS-Instanz wird vorerst auf dem Testplex untersucht. Danach wird in weiteren Schritten zuerst eine Db2 Datenbank und schließlich IBM MQ Queues dem Bereitstellungsprozess hinzugefügt. Um einen Testablauf der DATEV-Rechnungsschreibung mit der so generierten Laufzeitumgebung durchführen zu können, muss das Template auch in der Entwicklungsstange verfügbar sein. Ist dies sichergestellt wird der dadurch ermöglichte Bereitstellungsprozess anhand von drei use-cases aufgezeigt. Es folgt ein Fazit zu dieser Implementierung. Zuletzt folgt eine Bewertung der implementierten Provisionierungslösung durch die Stakeholder bei DATEV e.G. (Entwickler, Administration, Technologiestrategie).

5.1. Vergleich zwischen z/OSPT und z/OSMF

In folgender Tabelle 5.1 werden die beiden Tools an Hand folgender Kriterien miteinander verglichen:

- Schnittstelle
- Verwaltung von Templates
- Verwaltung von Instanzen bzw. Container
- Einsatz von Images

¹Siehe Absatz 1.2

Kriterium	z/OSPT	z/OSMF
Schnittstelle	Kommandozeile	browserbasierende Oberfläche
Verwaltung von Templates	Zuweisung von Domains und Tenants nicht intuitiv	Alle Arbeitsschritte intuitiv
Verwaltung von Instanzen bzw. Container	möglich	möglich
Einsatz von Images	Ja	Nein

Tabelle 5.1.: Vergleich zwischen z/OSPT und z/OSMF

Aus der Tabelle 5.1 ergibt sich folgendes Fazit:

z/OSPT ist durch den Einsatz von Images deutlich flexibler bezüglich der Konfigurationsmöglichkeiten der Templates. Jedoch ist die browserbasierende Schnittstelle von z/OSMF intuitiver als ein Kommandozeileninterface, dadurch fällt die Einarbeitung in automatisierte Bereitstellungsmechanismen einfacher. Hinzu kommt, dass innerhalb von z/OSPT die Zuweisung von Domains und Tenants nicht ohne weiteres möglich ist. Diese müssen in eine externe Konfigurationsdatei als Umgebungsvariablen aufgenommen werden.

Ausschlaggebender Grund für das Nutzen von z/OSMF für die Beantwortung der Forschungsfragen ist die browserbasierende Oberfläche.

5.2. Testplex

Der Zugriff auf Ressourcen und Tools bei DATEV e.G. wird über ein Rechtekonzept über RACF² verwaltet. Um die Forschungsfragen beantworten zu können, mussten vor Beginn der eigentlichen Untersuchung zunächst alle benötigten Rechte beantragt werden. Hierzu zählen unter anderem die Rechte für die Nutzung des Testplexes, die Nutzung von z/OSMF und z/OSPT und die Rechte für die Templateverwaltung innerhalb von z/OSMF. Beispielsweise benötigt „IBM Cloud Provisioning and Management for z/OS“ lesenden Zugriff auf den Speicherpfad der Template Dateien. Auf dem Testplex ist es möglich, die Rechte für das Erstellen der CICS Dateien, das Starten einer CICS Instanz und die Administration von Db2 und IBM MQ einer persönlichen UserID zu geben, was in der Entwicklungsstade nicht ohne weiteres umsetzbar ist.

Schließlich konnte, wie im Kapitel 3 beschrieben, mit dem ersten Versuch, das bei der Installation von z/OSMF mitgeliefertes „cics_getting_started“ Template zu provisionieren, begonnen werden. Ziel war es, mit dem Tool vertraut zu werden und die grundsätzliche Lauffähigkeit zu prüfen.

²Glossar ??

5.2.1. „cics_getting_started“-Template

Da es sich, wie in Kapitel 3 beschrieben, um ein mitgeliefertes Template handelt, sind alle benötigten Workflowdefinitionsfiles und Template Dateien vorhanden. Wie in Absatz 2.5.1 aufgezeigt, muss das Template in die Software Services von z/OSMF aufgenommen werden. Hierzu müssen die Template- und Workflow-Dateien in einem Unix Dateisystem auf dem Großrechner abgelegt sein. Nach dem Aufnehmen werden dem Template noch eine Domain und ein Tenant zugewiesen.³

Folgende Variablen sind in dem Variableinputfile nur mit Platzhaltern versehen und müssen für eine erfolgreiche Provisionierung ersetzt werden:

Variablenname	Kurzbeschreibung
DFH_REGION_APPLID	Applikations ID der zu provisionierenden CICS-Instance.
DFH_REGION_HLQ	High-level qualifier für die CICS Dateien.
DFH_STC_ID	User ID mit dem die CICS-Instanz startet.
DFH_REGION_VTAMNODE	Name des VTAM Knotens, wenn die CICS-Instanz hochfährt.
DFH_CICS_USSHOME	Homeverzeichnis des Unix System Services
DFH_CICS_HLQ	High-level qualifier von dem CICS Installationsort.

Tabelle 5.2.: Zu verändernde Variablen im „cics_getting_started“-Template

Als nächster Schritt wurde ein Testlauf und somit ein erster Versuch, das Template zu provisionieren, durchgeführt. Dabei kam es anfangs trotz Testplex-Umgebung zu Rechteproblemen, da die Anforderungen und Rahmenbedingungen der DATEV e.G. in dem standardisierten IBM Template natürlich nicht berücksichtigt waren, beispielsweise ist die Berechtigung für das Starten von Jobs von DATEV Vorgaben abhängig und CICS-Start-Mechanismen haben spezifische Anforderungen an die Eingabeparameter. Nach den notwendigen Anpassungen wurde das „cics_getting_started“-Template provisioniert und die definierten Aktionen aus der Aktiondefinitionsdatei getestet. Dabei wurde sich auf das Nutzen der z/OSMF Oberfläche fokussiert und nicht auf die eigentliche Funktionsfähigkeit der CICS-Instanz.

5.2.2. „cics_54“-Template

Wie in Kapitel 3 bereits beschrieben, ermöglicht das „cics_54“-Template komplexere Konfigurationsmöglichkeiten. Es mussten neben den in Tabelle 5.2 genannten Variablen noch

³Siehe 2.5

in Tabelle 5.3 folgende Variablen gesetzt werden. Die Kurzbeschreibungen und die Beschreibungen aller weiteren Variablen, die im Standard Template vorhanden sind, ist unter [\[http 201\]](http://201) zu finden.

Variablenname	Kurzbeschreibung
DFH_REGION_SEC	Legt fest, ob für das CICS Sicherheit im Allgemeinen aktiviert ist.
DFH_REGION_SECPRFX	Wenn DFH_REGION_SEC gesetzt ist, legt den Namen Prefix bei Authentificatio- nanfragen für Ressourcen fest.
DFH_LE_HLQ	High-level qualifier ⁴ für die Sprachumge- bung ⁵
DFH_REGION_LOGSTREAM	Legt fest, wie die Log Dateien für die provi- sionierte CICS-Instanz erstellt werden sol- len.
DFH_REGION_DFLTUSER	Default User ID für die CICS-Instanz.
DFH_REGION_MEMLIMIT	Dem CICS maximal zur Verfügung stehen- der Speicherplatz.
DFH_ZOS_PROCLIB	Datei auf dem Großrechner, die den Job enthält, der für das Erzeugen der CICS- Instanz zuständig ist.
DFH_ZOS_VSAM_VOLUME	Speichersystem auf welchem die Dateien ge- speichert werden sollen. Entscheidung kann auch an das System abgeben werden.

Tabelle 5.3.: Zu verändernde Variablen im „cics_54“-Template

Es wurden die gleichen Änderungen bezüglich der DATEV Job Vorgaben und der spezifischen Anforderungen der CICS-Start-Mechanismen wie in Absatz 5.2.1 durchgeführt. Nach Aufnahme in z/OSMF konnte die Provisionierung und Deprovisionierung erfolgreich durchgeführt werden. Dadurch wurden erste Erfahrungen mit z/OSMF und die grundsätzliche technische Funktionsweise in der echten DATEV e.G. Entwicklungsumgebung erprobt, was vor allem für die CICS-Administratoren einen wichtigen Schritt darstellte. Dennoch handelt es sich nur um eine IBM Standard CICS Instanz, die nicht mit einer DATEV e.G. spezifischen CICS Instanz zu vergleichen ist.

5.2.2.1. DATEV e.G. spezifischen CICS Template

Ziel dieses Schritts war die Provisionierung einer funktionsfähigen DATEV e.G. spezifischen CICS Instanz. In der im letzten Schritt provisionierten Standard IBM CICS-Instanz sind z.B. keine DATEV e.G. internen Transaktionen verfügbar. Dabei handelt es sich um interne Management-Transaktionen, mit denen z.B. Entwickler ihre fachlichen Transaktionen verwalten. Beispielhaft sei hier die Transaktion XMON zu erwähnen, ein DATEV proprietäres

Auskunftssystem über CICS-Transaktionen, Programme, Dateien usw.. Dies ist unabhängig von einer automatisiert provisionierten CICS-Instanz auch weiterhin notwendig.

Um dieses Template an die DATEV Umgebung anzupassen und letztendlich eine „DATEV CICS Instanz “ zu provisionieren, wurden folgende Schritte durchgeführt.

- Analyse des bestehenden Templates und der darauffolgenden Überarbeitung
- Umgang und Anpassung der CSD Datei
- Anpassung der Jobs und Skripte mit Schwerpunkt auf der „createCICS.jcl“-Datei.

Die Analyse ergab, dass das mitgelieferte „cics_54“-Template mit insgesamt 76 verwendeten Dateien sehr komplex und umfangreich ist. Es zählen alle Dateien, die direkt mit dem Template in Verbindung stehen. Im Zentrum des Templates steht die Workflow Definitionsdatei „provision.xml “ mit circa 583 Zeilen Code. In dieser sind alle Steps, die bei einer Provisionierung durchgeführt werden, definiert. Das Template beinhaltet nicht nur die Möglichkeit, CICS Instanzen mit unterschiedlichen Konfigurationen zu provisionieren, sondern auch, festzulegen, ob dies mit Skripten oder mit der REST-API geschieht.

Die Grundstruktur des „cics_54“-Templates wurde beibehalten, allerdings wurden nach dem „YAGNI“-Prinzip⁶ alle für eine DATEV e.G. spezifische CICS Instanz nicht benötigten Steps, die dazugehörigen Variablen und Dateien entfernt. Wie in Tabelle 5.4 zu sehen ist, konnten dadurch circa die Hälfte der Dateien gelöscht werden und bei der provision.xml wurde ein Drittel an Quellcode eingespart werden. Somit gewinnt das Template an Übersichtlichkeit und kann dadurch einfacher angepasst und gewartet werden. Das überarbeitete Template dient dem weiteren Vorgehen als Grundlage.

	IBM Standard CICS Template	DATEV e.G. spezifisches Template
Verwendete Dateien	76	36
provision.xml	circa 583 Codezeilen	circa 199 Codezeilen.

Tabelle 5.4.: Vergleich der beiden Templates im Bezug auf deren Umfang

Wie in Absatz 4.2.1 beschrieben, benötigt eine CICS Instanz spezifische Dateien. Die Namen dieser CICS Dateien wurden im dafür zuständigen Job an die DATEV e.G. internen Namenskonventionen angepasst.

Die nächste Voraussetzung für die Bereitstellung einer CICS Instanz ist das Erstellen einer CSD-Datei⁷. Folgende Entscheidung wurde in Zusammenarbeit mit dem CICS Administratorenteam getroffen. Um Auswirkungen auf bereits im Einsatz befindliche Instanzen zu

⁶ „You aren’t gonna need it“-Prinzip

⁷ Beschreibung siehe Absatz 4.2.1

verhindern, wird die von den Kollegen gepflegte CSD-Datei bei jeder Provisionierung kopiert und mit bestimmten Namenskonventionen gespeichert. So besitzt jede automatisch bereitgestellte CICS Instanz eine eigene CSD Datei. Durch dieses Vorgehen wird zudem sichergestellt, dass bei jeder Provisionierung die aktuellste Version der CSD Datei verwendet wird. Im Falle einer extra CSD Datei für alle provisionierten Instanzen müssten Änderungen, wie beispielsweise die Verfügbarkeit einer neueren CICS Version, an mehreren Stellen angepasst werden. Neue Ressourcen, wie zum Beispiel eine Verbindung zu einem Db2 Subsystem, können so ohne Nebenwirkungen zu anderen CICS Instanzen in die CSD aufgenommen werden. Ein weiterer Vorteil ist, dass bei der Deprovisionierung der CICS-Instanz diese Kopie der Standard Datei ohne Nebenwirkungen gelöscht werden kann.

Um dies umzusetzen, wurden zunächst zwei JCL Jobs geschrieben und in die entsprechende Workflowdefinitionfile eingebunden. Einer für die Implementierung des Kopiervorgangs und einer zum Löschen dieser Kopie. Für die Anpassung der CSD der zu provisionierenden Instanz ist ein weiterer Job notwendig. Es mussten bestimmte Gruppen (Zeilen sieben, neun und zehn in Abbildung 5.1) zu der CSD Liste der CICS Instanz hinzugefügt werden. Dabei handelt es sich um Ressourcen, die jede DATEV e.G. CICS Instanz benötigt. Die Reihenfolge ist relevant, da sie der Initialisierungsreihenfolge beim Startvorgang der CICS Instanz entspricht. Diese beiden Jobs wurde jeweils als neuer Step in den z/OSMF Workflow eingebunden.

```

1 //INIT EXEC PGM=DFHCSDUP
2 //STEPLIB DD DSN=CICS.TS54.SDFHLOAD,DISP=SHR
3 //DFHCSD DD DSN=CICS.DFHCSD.XPROV.TCICS42,DISP=SHR
4 //SYSPRINT DD SYSOUT=V
5 //*Reihenfolge ist WICHTIG!!
6 //SYSIN DD *
7 ADD LIST(TCICS42) GROUP(TESTPCT)
8 ADD LIST(TCICS42) GROUP(DB0C)
9 ADD LIST(TCICS42) GROUP(RCTTEST)
10 ADD LIST(TCICS42) GROUP(FCTT1)
11 ADD LIST(TCICS42) GROUP(MQPROV01)
12 //
```

Listing 5.1: Hinzufügen weiterer CSD Gruppen zur Liste der provisionierten CICS-Instanz mittels eines Jobs

Der abschließende Schritt zur Bereitstellung einer CICS Instanz ist das Erzeugen des STC Jobs. Die Definition bzw. das Script zur Erzeugung dieses Jobs ist in der „createCICS.jcl“-Datei zu finden. Im „cics_54“-Template beinhaltet diese ein Makro für die Validierung der SIT Parameter. Zusätzlich werden alle aus der Datei für die Eingabevariablen benötigten

Variablenwerte in temporäre Zwischenvariablen eingefügt. Danach folgt die Definition des Jobs, diese setzt sich aus folgenden Hauptbestandteilen zusammen:

- Einbindung der benötigten Bibliotheken
- Einbindung der zuvor angelegten CICS spezifischen Dateien
- Definition der SIT Parameter

Für das Einbinden der benötigten Bibliotheken und der zuvor angelegten CICS spezifischen Dateien ist nur das Hinzufügen weiterer DD-Statements notwendig.

In Abbildung 5.2 ist zu sehen, dass es vor allem bei der Definition der SIT Parameter zu tief verschachtelten if-Bedingungen kommen kann. Es handelt sich um den Code, der für das Einlesen der Variable „DFH_REGION_SITPARAMS“ aus der Eingabedatei zuständig ist. In dieser Variable werden die SIT Parameter als Komma separierter String angegeben. Für die Erzeugung eines DATEV e.G. spezifischen CICS wurde das ursprünglich in dem Template verwendete Makro für die Validierung von SIT Parametern beibehalten. Alles danach wurde zunächst durch eine zur Verfügung gestellten DATEV e.G. Standard JCL, für die Erzeugung eines CICS, ersetzt. Nach und nach wurde damit die für die DATEV eG spezifische CICS Provisionierung notwendige Logik, (siehe Abbildung 5.2), hinzugefügt. Damit wurde die vorher statische DATEV e.G. Standard JCL für das Erstellen von CICS Instanzen durch die Verwendung von Template Variablen flexibilisiert.

```

1 #set ($value5 = ${instance -DFH_REGION_SITPARMS})
2 #set ($multipart = "NO")
3 #set ($tempStr = "")
4 #if($value5 != "")
5 #foreach( $sit in $value5.split(","))
6 #if($multipart == "YES")
7 #if( $sit.indexOf(',') > 0 )
8 ## Validate SIT
9 #validateSit($tempStr.concat($sit.trim()))
10 #set ($multipart = "NO")
11 #else
12 #set ($tempStr = $tempStr + $sit.trim() + ",")
13 #end
14 #else
15 #if( $sit.indexOf('(') > 0 && $sit.indexOf(',') == -1 )
16 #set ($multipart = "YES")
17 #set ($tempStr = $sit.trim() + ",")
18 #else
19 #validateSit($sit.trim())
20 #end
21 #end
22 #end
23 #end

```

Listing 5.2: Setzen der SIT Parameter durch Auslesen der „DFH_REGION_SITPARAMS“ Variablen

Es wurden nur die wirklich benötigten SIT Parameter aufgenommen. Die anzunehmenden Werte wurden einzeln mit dem CICS Administratorenteam besprochen und festgelegt. Es ist zu beachten, dass es im IBM Standard Template zwei Möglichkeiten gibt, diese Parameter zu setzen. Für bestimmte SIT Parameter existiert eine Variable innerhalb des Templates. Für alle anderen ist die Variable „DFH_REGION_SITPARAMS“ vorgesehen. In dieser Arbeit wurde hauptsächlich mit letzterer Variante gearbeitet. Dadurch sind die SIT Parameter nur an einer Stelle im Template zu verwalten, beziehungsweise wird die Verwaltung nicht auf zwei Arbeitsweisen verteilt.

Durch die beschriebene Vorgehensweise wurde erfolgreich die Provisionierung eines DATEV e.G. spezifischen CICS Instanz umgesetzt. Getestet wurde die Nutzbarkeit der Instanz mit einem Anmeldevorgang an dieses CICS, wie in Abbildung 5.1 zu sehen ist. Darüber hinaus wurde erfolgreich nachgewiesen, dass Standard Transaktionen der DATEV e.G. in dieser Instanz funktionsfähig sind. Die Deprovisionierung verlief nach Plan.

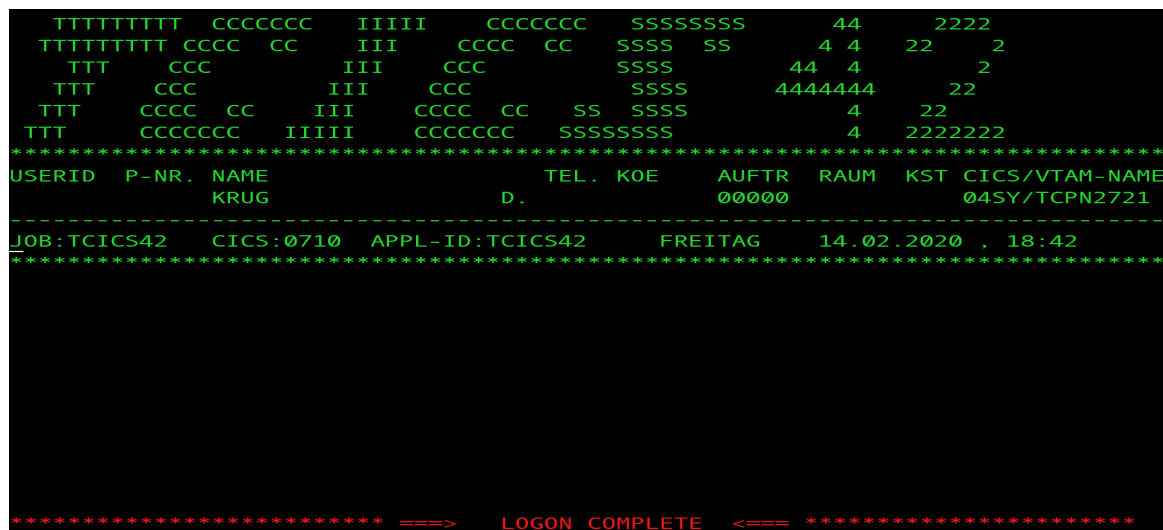


Abbildung 5.1.: Login Bildschirm der provisionierten DATEV spezifischen CICS-Instanz

5.2.2.2. Bereitstellung Db2

In diesem Absatz wird die Provisionierung einer Db2 Datenbank beschrieben. In der Systemumgebung Testplex bedeutet dies die Provisionierung der Datenbank ohne Tabelle und Daten.

Für die Erstellung einer Db2 Datenbank existiert innerhalb der DATEV e.G. eine REST-API. Wie im Absatz 2.5 beschrieben, ist es möglich, innerhalb eines Workflow Steps einen REST-Request abzuschicken. Der Code ist in Abbildung A.5 im Anhang zu finden. So muss im Body des Requests unter anderem der Datenbankname und eine UserID übergeben werden. Der Code für das Löschen der Datenbank sieht ähnlich aus, nur handelt es sich in diesem Fall um einen DELETE-Request. Die zwei notwendigen Steps wurden erzeugt und in den Workflow eingebunden.

Die API ist nur dazu fähig, Datenbanken auf einem bestimmten Datenbanksystem zu erzeugen. Um die Datenbank aus der CICS-Instanz heraus nutzen zu können, muss dem CICS dieses Datenbanksystem mitgeteilt werden. Hierfür ist, wie in Abbildung 5.1 in Zeile acht bereits zu sehen ist, das Hinzufügen einer weiteren CSD Gruppe notwendig ist, sowie die Aufnahme weiterer Bibliotheken in der „createCICS.jcl“. Dieser Aufruf wurde mittels neuer Variablen im Template möglichst dynamisch gestaltet und mussten in der Variableinputfile gesetzt werden.

5.2.2.3. Bereitstellung IBM MQ

In diesem Absatz wird die Provisionierung einer IBM MQ Queue im Testplex beschrieben. Es ist auch möglich einen IBM MQ Queue Manager zu provisionieren, der Fokus dieser

Arbeit liegt aber auf der Bereitstellung von Queues. Für die Bereitstellung eines IBM MQ Queue Managers bei DATEV e.G. ist laut IBM MQ-Administration vorerst keine automatische Bereitstellung vorgesehen, gegebenenfalls kann dies in einem zukünftigen Szenario umgesetzt werden. Ebenfalls in Abstimmung mit MQ- und CICS-Administration wurde entschieden, die Funktion eines Starts einer CICS Transaktionen über eine Queue vorerst nicht umzusetzen. Der Fokus lag auf der Prüfung, wie es möglich ist, eine einzelne Queue zu provisionieren, nicht die voll umfängliche Umsetzung der Anforderung der Anwendung DATEV-Rechnungsschreibung.

Die IBM stellt Programme für die Verwaltung und das Nutzen von Queues zur Verfügung. Diese können mittels eines Jobs und bestimmten Parametern gestartet werden. In Abbildung 5.2 ist die JCL des Jobs für das Erstellen einer Queue zu sehen. Das auszuführende Programm ist „CSQUTIL“ und als Parameter wird der Queuemanager übergeben. Unter dem DD Namen „MQSCIN“ ist der IBM MQ Befehl für das Erzeugen einer Queue zu sehen. Um zu Prüfen, ob die Queue auch funktionsfähig ist, wurde nach dem Erstellen auch mit Hilfe eines Jobs, eine Message auf die Queue geschrieben und wieder abgeholt. Der Job für das Löschen der Queues ist analog aufgebaut.

```

***** ***** Top of Data *****
000100 //P$PRVMQ1 JOB (0000,00000,00000,00000),KRUG,
000200 //          CLASS=V,MSGCLASS=V,TIME=(4,3),
000300 //          MSGLEVEL=(1,1),REGION=0M,NOTIFY=XXXXXXX,USER=YYYYYYY
000400 //*****
000500 //+
000600 //+  DISPLAY QUEUES
000700 //+
000800 //*****
000900 //STEPDISQ EXEC PGM=CSQUTIL,PARM='M00I',REGION=0M
001000 //+
001100 //STEPLIB DD DSN=MQS.TEST.LLT.SCSQAUTH,DISP=SHR
001200 //          DD DSN=MQS.TEST.LLT.SCSQANLE,DISP=SHR
001300 //SYSPRINT DD SYSOUT=*
001400 //SYSIN DD +
001500 COMMAND DDNAME(MQSCIN) FAILURE(STOP)
001600 //+
001700 //MQSCIN DD +
001800 DEFINE QLOCAL(AWTB.PABHGMSHARE) -
001900 REPLACE -
002000 MAXDEPTH(6) -
002100 PROCESS(AWTP.AWTP) -
002200 TRIGGER -
002300 MAXMSGL(1300) -
002400 INITQ(SERVICE.TCICS42.INITQ) -
002500 TRIGTYPE(EVERY) -
002600 QDEPTHHI(80) -
002700 QDEPTHLO(40)
002720 //+
002800 //PUT EXEC PGM=CSQ4BCS2,
002900 //          PARM=('AWTB.PABHGMSHARE M00I')
003100 //STEPLIB DD DSN=MQS.TEST.LLT.SCSQAUTH,DISP=SHR
003200 //          DD DSN=MQS.TEST.LLT.SCSQANLE,DISP=SHR
003300 //          DD DSN=MQS.TEST.LLT.SCSQLOAD,DISP=SHR
003400 //STDOUT DD SYSOUT=*
003500 //STDERR DD SYSOUT=*
003600 //SYSPRINT DD SYSOUT=*
003700 //SYSIN DD +
003800 TEST
003900 //+
004000 //GET EXEC PGM=CSQ4BCJ1,
004100 //          PARM=('M00I AWTB.PABHGMSHARE 1 D N')
004300 //STEPLIB DD DSN=MQS.TEST.LLT.SCSQAUTH,DISP=SHR
004400 //          DD DSN=MQS.TEST.LLT.SCSQANLE,DISP=SHR
004500 //          DD DSN=MQS.TEST.LLT.SCSQLOAD,DISP=SHR
004600 //SYSDBOUT DD SYSOUT=*
004700 //SYSABOUT DD SYSOUT=*
004800 //SYSPRINT DD SYSOUT=*
004900 //SYSOUT DD SYSOUT=*
005000 //
***** ***** Bottom of Data *****

```

Abbildung 5.2.: Define IBM Queue, am Beispiel einer Trigger Queue

Ähnlich wie in Absatz 5.2.2.2 für die Datenbank-Provisionierung beschrieben, muss der CSD Datei eine weitere Gruppe für den Queuemanager angegeben werden. Zu sehen in

Abbildung 5.1 in Zeile 11. Dadurch hat die CICS-Instanz Zugriff auf alle Queues, die sich innerhalb dieses Managers befinden. Des Weiteren ist die Aufnahme weiterer Bibliotheken in der „createCICS.jcl“ notwendig.

5.3. Entwicklungsstage

Innerhalb der Entwicklungsstage sind die Sicherheits- und Rechtsvorschriften schärfer als auf dem Test-Plex. So wäre es zwar möglich, alle für die administrativen Aufgaben notwendigen Rechte einer persönlichen UserID zu geben. Dies würde bedeuten, dass alle Anwender dieses Templates diese Rechte auch benötigen. Damit bestünde eine potentielle Gefahr für das System, da sie damit auch außerhalb des Templates diese Rechte besitzen würden. Somit wurde in Absprache mit den Administratorenteams für CICS und IBM MQ festgelegt, hierfür jeweils einen technischen User⁸ zu beantragen. Diesem werden nur die für das Template benötigten Rechte übergeben und er ist somit use-case-spezifisch. Um als Anwender das Template nutzen zu können, werden nur die Rechte benötigt, Jobs mit diesen technischen Usern ausführen zu dürfen. Für Db2 ist ein solcher User nicht notwendig, da das Datenbanksystem hinter der REST-API für alle zugänglich ist und jeder darauf Datenbanken erstellen darf.

Bei der Übertragung des Templates vom Test-Plex in die Entwicklungsstage waren Anpassungen in allen drei Bereichen des Templates notwendig.

5.3.1. CICS Anpassung

Dass der CICS spezifische technische User zum Einsatz kommt, musste der „Job“ Baustein jeder JCL in jedem Step modifiziert werden. Dafür bietet z/OSMF die Möglichkeit beim Zuweisen des „Tenants“ eine Standard Jobkarte, die vor jeden Job des Templates eingefügt wird, zu hinterlegen. Die CICS spezifischen Dateien können von der täglichen Datensicherung der Entwicklungsstage ausgeschlossen werden. Da diese bei der Deprovisionierung gelöscht werden. Um dies zu gewährleisten musste der Messageclass Parameter mit dem Wert „NONE“ angegeben werden.

Außerdem wird die CSD Datei, die als Vorlage gilt, durch die Standard Entwicklungsstage CSD Datei ersetzt. In der Entwicklungsstage kommen im Vergleich zum Testplex andere Db2 und IBM MQ Bibliotheken zum Einsatz. Dahingehend wurde die „createCICS.jcl“-Datei angepasst. Zusätzlich musste ein SIT Parameter angepasst werden, so dass die Log Dateien funktionisfähig sind. Eine weitere CSD Gruppe musste hinzugefügt werden. Siehe

⁸User ID mit zunächst keinen Berechtigungen

Zeile 16 im Codeabschnitt 5.3. Diese sorgt dafür, dass die Bibliotheken, die die kompilierten Programme der kompletten Entwicklungsstage beinhalten, zur Verfügung stehen. Außerdem kam noch eine neue Bibliothek hinzu. Diese dient später als Ablageort der kompilierten Programme, die explizit nur in diese CICS-Instanz vorhanden sind. Dies ist ein Standardvorgehen innerhalb der DATEV e.G. um neue Programmversionen zu testen.

5.3.2. Db2 Anpassung

Eine genauere technische Analyse der DATEV-Rechnungsschreibungsdatenbank kam zu dem Ergebnis, dass es zwar möglich wäre diese Datenbank zu provisionieren, dies aber den zeitlichen Rahmen dieser Arbeit übersteigen würde. Der Grund hierfür ist die Komplexität der benötigten Tabellen. So wird auf drei Tabellen für die Ermittlung der Produktstammdaten lesend zugegriffen, auf neun weitere bei der Bestimmung der Preisabhängigkeiten. Auf die Tabellen wird nicht direkt zugegriffen, sondern über Views⁹. Bei den meisten werden innerhalb der View noch weitere Tabellen, teilweise aus anderen Datenbanken, gejoint. Insgesamt besteht das System aus 14 Tabellen, die auf vier Datenbanken aufgeteilt sind, und 12 Views für den Zugriff auf diese Tabellen.

Die Db2 Administration muss dafür Vorarbeit leisten. Mit dieser wurde begonnen, jedoch stellte sich heraus, dass die Komplexität (circa 600 Zeilen Code¹⁰ für einen kleinen Teil an Tabellen) der Anwendung DATEV Rechnungsschreibung im Rahmen dieser Arbeit als zu umfangreich angesehen wurde. Sollte sich die Provisionierung generell als zielführend erweisen wird dieser Einmalaufwand erbracht werden.

Für die weitere Arbeit werden Datenbanken, die in einem anderen Datenbanksystem bereits vorhanden sind, genutzt. Hierfür mussten die dafür vorgesehenen Variablen in der Eingabedatei des Templates angepasst werden. Dadurch ändert sich die Gruppe in Zeile acht im Codeabschnitt 5.1 von „DB0C “ auf „DB0T “. Außerdem wurden sowohl in der Provisionierungs- als auch in der Deprovisionierungsdatei die Datenbanksteps auskommentiert und somit kommen diese nicht mehr zum Einsatz.

5.3.3. IBM MQ Anpassung

Da für die DATEV Rechnungsschreibung, wie im Absatz 4.3.2 beschrieben, sehr viele gleichartige Queues benötigt werden, wurde für die Erstellung dieser von den IBM MQ Administratorenteam ein REXX Skript angefertigt. Dies geschah unabhängig dieser Arbeit zum Zeitpunkt der Einführung des aktuellen DATEV Rechnungsschreibungsprozesses. Dieses

⁹Alias eines Datenbankabfrage, auf die wie auf eine normale Tabelle zugegriffen werden kann

¹⁰Data Definition Language im Anhang A.3 zu finden

Skript steht dieser Arbeit zur Verfügung. Für die Provisionierung IBM MQ Queues waren folgende Arbeitsschritte notwendig.

- Anpassung des zur Verfügung stehenden Skriptes
- Implementierung von Jobs für restliche Queues
- Anpassung der CICS CSD Datei

Hierfür wurden zunächst die Eingabeparameter durch vorher angelegte Templatevariablen ersetzt. Diese steuern, wie viele Queues jeweils angelegt werden, auf welchen Queue Manager die Queues angelegt werden und den ersten Qualifier des Queuenamens. Für den restlichen Queuenamen existiert auch eine Variable, in dieser werden die Namen als Komma separierte Liste angegeben und ausgelesen. Anhand dieser Namen wird dann die maximale Queue-tiefe und die maximale Länge einer einzelnen Nachricht festgelegt. Im alten Skript wurden die Queues mit Hilfe einer Queue, die als Vorlage dient, angelegt. Im Fall einer Provisionierung kann nicht davon ausgegangen werden, dass diese Vorlagen zur Verfügung stehen. Deshalb wurden die benötigten Parameter explizit manuell angegeben. Um die damit erstellten Queues zu testen, wurde eine Routine entwickelt, die eine Nachricht auf die Queue schreibt und diese wieder abholt. Anschließend wurde das Skript in den Provisionierungsworkflow mit Hilfe eines neuen Steps aufgenommen.

Für die Deprovisionierung der Queues besteht noch kein Skript. Als Grundlage kann das vorher angepasste Provisionierungsskript dienen. Hierfür musste der „Define“-Befehl für die Erstellung von Queues durch den „Delete“-Befehl ausgetauscht werden. Die Logik für die Ermittlung der maximalen Queue-tiefe und der maximalen Nachrichtenlänge wird dafür nicht mehr benötigt und konnte entfernt werden.

Die durch die beiden Skripte erstellten Queues sind nur für den Datenaustausch zwischen der CICS Transaktion für die Preisermittlung und dem Batch Ablauf zuständig. Wie in Absatz 4.3.2 beschrieben, benötigt der Ablauf noch weitere Queues. Da es sich hierbei um spezielle Queues handelt, wurde auf die im Absatz 5.2.2.3 gezeigte Technik zurückgegriffen. Bei der Antwort-Queue für die Ermittlung der Listenpreise handelt es sich um eine Queue ohne besondere Parameter. Es werden noch zwei Trigger-Queues benötigt, die über Prozesse eine Transaktion im CICS starten. Als letzter Baustein für das Triggering der Transaktion wird noch eine Initiation Queue benötigt. Diese muss im CICS hinterlegt sein.

Jeder CICS-Instanz kann nur eine Initiation Queue zugewiesen sein. Dadurch benötigt jedes CICS eine eigene Initiation Queue. Die Zuweisung geschieht in der IBM MQ CSD Gruppe. Somit müsste für jede provisionierte CICS-Instanz im Voraus eine solche CSD Gruppe angelegt werden. In Absprache mit der IBM MQ-Administration wurde entschieden, die Verwaltung der IBM MQ CSD Gruppe komplett dem Template zu übergeben. Diese Entscheidung hatte eine Änderung des in Abbildung 5.1 gezeigten Codes zur Folge. So wird,

wie in Abbildung 5.3 dargestellt, zunächst eine Gruppe angelegt und erst anschließend dem CSD hinzugefügt.

```

1 //INIT EXEC PGM=DFHCSDUP
2 //STEPLIB DD DSN=CICS.TS54.SDFHLOAD,DISP=SHR
3 //DFHCSD DD DSN=CICS.DFHCSD.XPROV.TCICS42,DISP=SHR
4 //SYSPRINT DD SYSOUT=V
5 //*Reihenfolge ist WICHTIG!!
6 //SYSIN DD *
7 DEFINE MQCONN(M00I)
8     G(MQPROV01)
9     MQNAME(M00I)
10    INITQ(SERVICE.TCICS42.INITQ)
11 ADD LIST(TCICS42) GROUP(TESTPCT)
12 ADD LIST(TCICS42) GROUP(DB0T)
13 ADD LIST(TCICS42) GROUP(RCTTEST)
14 ADD LIST(TCICS42) GROUP(FCTT1)
15 ADD LIST(TCICS42) GROUP(MQPROV01)
16 ADD LIST(TCICS42) GROUP(RPL)
17 //
```

Listing 5.3: Erstellung einer neuen CSD Gruppe

Für jeden IBM MQ bezogenen Job wurde zuallerletzt die Jobkarte angepasst und der technische User der CICS-Administration durch den technischen User der IBM MQ-Administration, der für administrative Aufgaben berechtigt ist, ausgetauscht.

5.3.4. Testablauf

Für die Prüfung der Funktionsfähigkeit der so generierten Laufzeitumgebung steht dieser Arbeit ein Testablauf zur Verfügung. Dieser wurde von den Mitarbeitern der DATEV Rechnungsschreibung beigesteuert. Dabei handelt es sich um einen Teilablauf des gesamten DATEV Rechnungsschreibungsprozesses. In diesem Ablauf wird nur die Preisermittlung, die die Laufzeitumgebung CICS benötigt, getestet. Als Eingabe dienen vordefinierte Dateien und die Ergebnisse werden ebenfalls in Dateien geschrieben. Der Ablauf liegt in Form von zwei Jobs vor. Beide sind in der gleichen JCL Datei definiert, somit starten beide zeitgleich. Dies ist notwendig, da der erste Job die Verarbeitung im CICS über die Queues startet und der zweite auf die Ergebnisqueues lauscht.

Um den Ablauf auch auf der vorher provisionierten Laufzeitumgebung zu starten, musste lediglich der verwendete Queue Manager angepasst werden. Über die Queues und das ver-

wendete Triggering wird die Transaktion im richtigen CICS gestartet. Um die Ausgabe zu prüfen wurde der gleiche Testablauf mit den gleichen Eingabedateien auf der für Testzwecke üblichen Laufzeitumgebung durchgeführt. Anschließend wurden die Ausgabedateien beider Läufe verglichen.

5.4. Bereitstellungsprozess aktuelles Template

Bei dem Bereitstellungsprozess, der durch das aktuelle Template möglich gemacht wird, sind drei Fälle zu unterscheiden:

1. Use-Case: Neue Template Instanz

Dem Entwicklerteam steht das Template in z/OSMF zur Verfügung und es wurde noch keine Instanz dieses Templates provisioniert. Es wird eine neue Instanz benötigt.

2. Use-Case: Zusätzliche Template Instanz

Dem Entwicklerteam steht das Template in z/OSMF zur Verfügung und es steht bereits eine Instanz dieses Templates zur Verfügung. Es wird eine weitere Instanz benötigt.

3. Use-Case: Änderungen durch Administratorenteam

Das Administratorenteam führt Änderungen an einer Workflow Definitionsdatei durch. Hier ist zwischen zwei weiteren Fällen zu unterscheiden:

- a) Das Template wurde noch nicht veröffentlicht.
- b) Das Template wurde veröffentlicht.

5.4.1. Use-Case: Neue Template Instanz

Der Mitarbeiter meldet sich an der zOSMF Oberfläche an und klickt auf den Menüleistepunkt „Cloud Provisioning“. Anschließend öffnet er die „Software Services“ und wählt dort das oben genannte Template aus. Er kann es ohne Änderungen provisionieren und damit seine Programmabläufe testen.

5.4.2. Use-Case: Zusätzliche Template Instanz

Mit dem aktuellen Stand muss der Mitarbeiter wissen, an welchem Speicherort das Template abgelegt ist, da er die Template - nicht die Workflowdateien - kopieren muss. Es sind Änderungen der Variableinputfile notwendig. Unter anderem ist eine andere CICS Application ID zu wählen. Um die Queues und IBM MQ Prozesse aus Fall eins nicht zu überschreiben, muss ein anderer Queue Manager gesetzt werden. Dieser Queue Manager muss von

den zuständigen Administratorenteam manuell bereitgestellt werden. Die Erzeugung einer von Fall eins unabhängigen Instanz setzt die Aufnahme eines neuen Templates, welches die veränderten Dateien beinhalten, in z/OSMF voraus.

5.4.3. Use-Case: Änderungen durch Administratorenteam

Ein Template ist dann veröffentlicht, wenn es den berechtigten Teams zur Verfügung steht. Zunächst muss der Speicherort der zu bearbeiteten Dateien bekannt sein. Anschließend kann die Änderung mit einem Editor nach Wahl durchgeführt werden.

5.4.3.1. nicht veröffentlichtes Template

Hier kann das Template in der z/OSMF Oberfläche per Mausklick aktualisiert werden.

5.4.3.2. veröffentlichtes Template

Um die Funktionsfähigkeit der veralteten Instanzen weiterhin sicherzustellen, muss eine neue Version des Templates erzeugt werden. Dies ist auch per Mausklick zu lösen.

5.5. Fazit Realisierung

Am Ende der Realisierung steht ein funktionsfähiges Template. Dieses Template provisioniert ein CICS und die benötigten IBM MQ Queues. Wie in Absatz 5.3.2 beschrieben, wurde eine Db2 Datenbank wegen hoher Komplexität außen vorgelassen. Auf dem Testplex wurde bewiesen, dass die Provisionierung einer Datenbank möglich ist. Des Weiteren wäre die Provisionierung von Tabellen mit hohem einmaligen Arbeitsaufwand ebenfalls möglich. Ein Testablauf der Beispielanwendung DATEV Rechnungsschreibung in einer provisionierten, isolierten CICS-Laufzeitumgebung konnte korrekt durchgeführt werden.

Folgende Probleme wurden im Rahmen der Implementierung erkannt:

- Nicht sprechende Fehlermeldungen von z/OSMF
- Nicht identifizierbare Programmiersprache
- Nicht optimales Zugriffsrechtekonzept

Als erstes Problem sind nicht sprechenden Fehlermeldungen von z/OSMF, Abbildung 5.3, zu nennen. z.B. wird bei dem Hinzufügen und Aktualisieren eines Templates in z/OSMF das Template und damit alle davon benötigten Dateien auf Syntaxfehler geprüft. Die in


 During template evaluation, one or more errors were detected in the workflow definition file.

Abbildung 5.3.: Beispiel einer Fehlermeldung von zOSMF

Abbildung 5.3 gezeigte Meldung tritt dann ein, wenn ein solcher Syntaxfehler vorhanden ist. Es ist aber nicht zu erkennen, welcher Fehler genau vorliegt, noch nicht einmal in welcher Datei dieser auftritt. Zudem auch keine genaue Anzahl an auftretenden Fehlern. Dieser Umstand, kombiniert mit 36 bestehenden Dateien, erschwert die Fehlersuche. Im Gegensatz dazu wird im Fehlerfall beim Ausführen eines Steps immer der Fehlercode und der genaue Ort des Fehlers ausgegeben. Beispielsweise wird bei einem Step, in dem ein REST Aufruf durchgeführt wird, und ein Fehler auftritt, der Requestcode und die hinterlegte Fehlermeldung an der z/OSMF Oberfläche angezeigt.

Ein weiteres Problem ist eine nicht genau identifizierbare Programmiersprache, die für die dynamische Generierung von Skripten genutzt wird. So ermöglicht diese die dynamische Wertzuweisung von zum Beispiel REXX-Variablen durch Variablen des Templates. Außerdem besteht eine Art von String Verarbeitung. Zu beachten ist, dass wenn am Zeilenanfang ein „#“ steht, kann diese Programmiersprache verwendet werden. In Abbildung 5.4 ist ein Beispiel zu sehen. Dort werden die Queuenamen, die als kommaseparierte Liste in der Templatevariable „DFH_MQ_QUEUE NAMES“ angegeben sind, ausgelesen und in eigenen REXX Variablen gespeichert. Zu sehen ist zunächst eine „set“ Anweisung, mit der Variablen zugewiesen werden können, If-Bedingungen und eine foreach-Schleife stehen außerdem zur Verfügung.

```

1 i=0
2 #set ($names = ${instance-DFH_MQ_QUEUE NAMES})
3 #set ($multipart = "NO")
4 #set ($tempStr = "")
5 #if($names != "")
6 #foreach( $queue in $names.split(", "))
7 i=i+1
8 names.i="$queue "
9 #end
10 #end
11 names.0=i

```

Listing 5.4: Auslesen der „DFH_MQ_QUEUE NAMES“ Variablen und schreiben in REXX Variablen

In Abbildung 5.5 wird das Ergebnis, welches zur Laufzeit ausgeführt wird, dargestellt. Es ist zu erkennen, dass nur noch die für das REXX Skript notwendigen Codeabschnitte vorhanden sind. Dadurch können sehr dynamische Templates erstellt werden. Jedoch wurde weder

eine Dokumentation zu dieser Sprache, noch um welche Sprache es sich genau handelt gefunden. Somit liegt dem Wissen über diese Sprache nur der Code aus Beispielen der IBM zu Grunde.

```

1  i=0
2  i=i+1
3  names.i="L1.GPNRBERINFO"
4  i=i+1
5  names.i="L1.KLAMMERINFOLIST"
6  i=i+1
7  names.i="L1.KUNDENPREISINFOLIST"
8  i=i+1
9  names.i="L1.PABHREFERENZLIST"
10 names.0=i

```

Listing 5.5: Zur Laufzeit erzeugtes Skript, der Grundlage aus Codeabschnitt 5.4

Ein weiterer Problempunkt ist das mit z/OSMF und dem Template einhergehende Zugriffsrechtekonzept. Die z/OSMF Berechtigungsgruppen sind nicht an die DATEV e.G. internen Richtlinien angepasst. Die Aufnahme in eine solche Gruppe, um zum Beispiel die z/OSMF Oberfläche nutzen zu dürfen, geschieht auf Zuruf und manuelles Hinzufügen einer User ID durch einen Mitarbeiter. Außerdem ist der Einsatz einer für das ganze Template gültigen Standard Jobkarte, um technische User verwenden zu können, nicht optimal. z/OSMF bietet hier eigentlich eine Möglichkeit in der Stepdefinition einen „runAsUser“ anzugeben. Unter diesem User würde der Step dann ausgeführt werden. Folglich ist das die Stelle an der zum Beispiel für CICS Steps der technische User für administrative CICS Aufgaben angegeben werden müsste. So würde das Gewähren der expliziten Rechte zum Starten eines Jobs mit der technischen User Id entfallen und damit die manuelle Arbeit des „Gewährens“, was mittels eines Formulars beantragt wird. Jedoch um einen „runAsUser“ in der Stepdefinition angeben zu können, muss in der dem Template zugewiesenen „Domain“ ein sogenannter „Cloud Security Admin“ hinterlegt sein. Dieser würde sicherstellen, dass nur die für ein Template zugelassenen User dieses Template auch provisionieren dürfen. In dieser Arbeit wird die mitgelieferte „Default Domain“ genutzt, in dieser ist kein „Cloud Security Admin“ angegeben. Da es sich um die Standard „Domain“ handelt, darf diese nicht geändert werden. Somit müsste eine eigene „Domain“ angelegt werden um einen „Cloud Security Admin“ hinterlegen zu können. Dadurch, dass sich z/OSMF bei der DATEV e.G. noch in einem Teststadium befindet, wird von der Erstellung einer eigenen „Domain“ abgesehen. Dies ist der Grund für den nicht optimalen Einsatz der oben genannten Jobkarten. An diesen beiden Fällen ist zu erkennen, dass das Rechtekonzept noch nicht für einen firmenweiten Einsatz ausgelegt ist und noch überarbeitet und angepasst werden muss. Dies ist jedoch explizit nicht Bestandteil dieser Arbeit.

5.6. Interviews

Die Fragebögen werden im Folgenden zunächst nach Gruppen ausgewertet. Schließlich wird daraus ein allgemeines Stimmungsbild abgeleitet.

5.6.1. CICS Administratoren

Der momentan etablierte Bereitstellungsprozess wird von der CICS Administration mit hohem manuellen Aufwand verbunden. Dies kombiniert mit viel Abstimmungsbedarf zwischen den Administratoren- und Entwicklerteams führt dazu, dass der Prozess als langsam und verbesserungswürdig angesehen wird. In der Umsetzung mit z/OSMF sieht die CICS Administration trotz des vermuteten hohen Einarbeitungsaufwandes bereits einen Mehrwert. Der Hauptvorteil des vorgestellten z/OSPT Lösungsansatzes sei dessen Flexibilität. Jedoch schreckt die dadurch benötigte Komplexität des zu erstellenden dynamischen Templates ab. Dieser Effekt wird durch fehlende Toolunterstützung und dem dadurch fehlenden Syntaxhighlighting beim Editieren der Template Dateien bzw. der Workflowdefinitionfiles verstärkt. Nach der Hürde des Einarbeitungsaufwandes und Eingewöhnung in das Editieren von Template Dateien und der Workflowdefinitionsdateien stehe einer aufwandssparenden Provisionierung mittels des „IBM Cloud Provisioning and Management for z/OS“-Toolkits nichts im Wege.

5.6.2. Db2 Administratoren

Das ganze „IBM Cloud Provisioning and Management for z/OS“-Toolkit wird als sehr mächtig, aber komplex beschrieben. Im Vergleich dazu funktioniere der momentan etablierte Bereitstellungsprozess sehr gut, da dieser bereits lange eingesetzt wird. Jedoch könnten lange Wartezeiten, die durch die vielen Abhängigkeiten zwischen Personen und Abteilungen zu Stande kommen, durch einen automatisierten Ablauf mittels des Toolkits eliminiert werden. Der durch z/OSMF ermöglichte Prozess zeige zwar das eine Automatisierung in diesem Bereich möglich ist, aber auch das noch viel Forschungsaufwand und Weiterentwicklung in diesem Bereich notwendig ist, um die Provisionierung wirklich nutzen zu können. z/OSPT diene dabei als Hilfsmittel den Bereitstellungsprozess in eine CI/CD-Pipeline aufzunehmen und so weiter zu automatisieren. Das durch das Toolkit ermöglichte automatisiertes Deployment von z/OS Middleware wird als notwendiger Schritt, um den Mainframe weiterhin erfolgreich zu betreiben, betrachtet.

5.6.2.1. IBM MQ Administratoren

Die IBM MQ Administratoren stimmen überein, dass der momentan etablierte Bereitstellungsprozess mit einem hohen manuellen Arbeitsaufwand verbunden ist. Durch Arbeiten auf Zuruf und Kommunikation über Telefon, Email oder Terminen entstehen häufig Rückfragen. Die Meinungen über die Lösungen mit z/OSMF und z/OSPT sind jedoch unterschiedlich. So biete der z/OSMF Ablauf zwar einem Mehrwert durch Abbau von manuellen Eingriffen, allerdings sei dieser bezogen auf die Queues noch sehr spezifisch. Um einen größeren Mehrwert zu generieren, ist das automatische Provisionieren eines Queuemanagers mit in das Template aufzunehmen. Hier sei der zusätzliche Arbeitsaufwand nicht zu vernachlässigen. Beim z/OSPT Lösungsansatz wird kritisiert, dass es sich nur um „pseudo“-Docker Container handle. Die hier von der IBM gewählte Namensgebung führt zur Verwirrung, da ein z/OSPT Container zwar ein Container im Sinne von einem Behälter für Middleware ist, jedoch nicht im Sinne eines Docker Containers, der in unterschiedlichen Systemumgebungen lauffähig ist. Ein weiterer Kritikpunkt ist, dass das gesamte Toolkit im Vergleich zu Jenkins nicht einfach genug zu verwenden sei. Wenn diese Probleme behoben werden können, könnten sich die IBM MQ Administratoren vorstellen IBM MQ Ressourcen mittels des Toolkits zu verwalten. Dabei sei zu beachten, dass erst noch eigene Erfahrungen mit dem Toolkit gesammelt werden sollten, bevor eine endgültige Bewertung möglich ist.

5.6.2.2. Entwicklerteam der DATEV Rechnungsschreibung

Der Entwicklerfragebogen wurde zusammen mit zwei Entwicklern ausgefüllt.

Aus Sicht des Entwicklers wird vor allem für den in Absatz 5.4.2 beschriebenen Fall viel Wissen über die z/OSMF Oberfläche und das Template selbst benötigt. Dieses Wissen müsse auch bei nicht häufiger Nutzung über einen längeren Zeitraum erhalten werden. Deshalb sei der z/OSPT Lösungsansatz, mit dem auf die z/OSMF Oberfläche durch den Einsatz mittels Jenkins oder dem DATEV „Marktplatz“ verzichtet werden kann, besser geeignet. Ist diese Integration möglich wird ein Hauptvorteil darin gesehen, dass der Bereitstellungsprozess mehr in den Händen des eigenen Teams liegt. So sei eine Steigerung der Effizienz möglich. Es stehe dem Sammeln von Erfahrungen mit dem Prozess und dem kompletten Toolkit nichts im Wege. Für die Zukunft könne sich die Nutzung auch für die Qualitätsicherungs- und Produktionsstage, um dort beispielsweise CICS-Instanzen horizontal zu skalieren, vorgestellt werden.

5.6.2.3. Fachberaterin im Bereich Technologiestrategie

Laut der Fachberaterin im Bereich Technologiestrategie ist der gezeigte Ablauf beziehungsweise die z/OSMF Oberfläche für die Aufgabe des Provisionierens von z/OS Middleware

geeignet. Jedoch sei es besser wenn z/OSMF in den bereits existierenden „Marktplatz“ für DATEV Cloud Lösungen integriert wäre. Der Prozess, der mit Hilfe von z/OSPT ermöglicht wird, wird als gut angesehen, da durch ihn die Entwicklung von z/OS Anwendungen an die Vorgehensweise der Cloud Native Entwicklung angenähert wird. Hier kommt die Rolle des Build Engineers auch für solche Anwendungen ins Spiel. Dieser kümmert sich um die Erstellung und Pflege der Build-Pipeline. Große Nachteil im momentan etablierten Bereitstellungsprozess sei vor allem, dass eine Anzahl von Entwicklern, die parallel an einem Produkt arbeiten, sich die gleiche Entwicklungsumgebung teilen. So arbeiten alle mit der gleichen CICS-Instanz, der gleichen Test-Datenbank und mit den gleichen IBM MQ Queues. Dadurch beeinflussen Änderungen des einen Entwicklers die Tests der anderen Kollegen, es entsteht Koordinationsaufwand. Falls Änderungen an der Umgebung notwendig sind, kann während dieser Zeit kein Entwickler weiterarbeiten. Hier liege der Vorteil des „IBM Cloud Provisioning and Management for z/OS“-Toolkits. Es ermögliche aus Entwicklersicht eine sehr einfache, schnelle Möglichkeit eine isolierte Umgebung bereitzustellen, unabhängig von den Administratorenteams. Zusätzlich diene die Konfigurationsdateien auch als Dokumentation, welche Ressourcen für ein erneutes Erstellen der Umgebung notwendig sind.

Abschließend lässt sich sagen, dass aus Sicht einer Fachberaterin im Bereich Technologie-strategie dieses Toolkit die Entwicklung beziehungsweise den Bereitstellungsprozess deutlich verbessern könne. So sei für den Entwickler ein an die Cloud Native Welt angenäherter Entwicklungsprozess möglich. Dadurch wird der Wechsel zwischen beiden Umgebungen immer fließender.

5.6.3. Meinungsbild

Über alle Gruppen hinweg lassen sich folgende Punkte zusammenfassen:

- neuer Prozess notwendig
- z/OSPT Lösung bevorzugt
- erste Erfahrungen sammeln

Es stimmen alle Gruppen überein, dass der momentan etablierte Bereitstellungsprozess für Mainframesubsysteme durch viele Absprachen und Abstimmungsaufwand zeitaufwändig ist. Sie würden einen automatisierten und dadurch schnelleren und weniger fehleranfälligen Prozess begrüßen.

Jedoch muss dieser Prozess aus Entwicklersicht mit minimalem Konfigurationsaufwand verbunden sein. Dies könnte durch eine Provisionierung mittels z/OSPT und einer Integration in eine Jenkins Build Pipeline oder durch die Einbindung in den „DATEV Marktplatz“

mittels eines entwickelten „Service Brokers“ gewährleistet werden. Aus Sicht der Administratoren sind mit dieser Umsetzung nur wenige allgemeine Templates zu verwalten, da die Entwickler mit z/OSPT Images arbeiten und keine weiteren Templates erzeugen. Um diese Punkte zu ermöglichen, muss das Template umgestaltet werden. Der dadurch in den Administratorenteams entstehende Aufwand und die damit verbundene steile Lernkurve hat eine abschreckende Wirkung.

Trotz dieser abschreckenden Wirkung sind auch die Administratorenteams bereit, falls die Kapazitäten vorhanden sind, den Bereitstellungsprozess mit Hilfe des „IBM Cloud Provisioning and Management for z/OS“ zu verbessern. Aus Sicht der Technologiestrategie ist dies ein wichtiger und notwendiger Schritt hin zu einem Cloud Native ähnlichen Prozess.

Kapitel 6.

Ausblick

Je weiter sich das Projekt der vorliegenden Arbeit dem Abschluss näherte desto mehr kristallisierte sich ein Hauptproblem heraus. Das erstellte Template ist sehr auf die DATEV Rechnungsschreibung spezialisiert, das heißt, es ist funktionsfähig, kann aber nicht ohne zeitaufwändige Eingriffe in das Template, in die Workflowdefinitionsdateien und die eigentlichen REXX Skripte und Jobs, an eine andere Anwendung angepasst werden. Folglich müsste das Template dynamischer implementiert sein. Um dies zu verdeutlichen, wird als Beispiel die Provisionierung von IBM MQ Queues herangezogen. Momentan werden die Prozesse und die Trigger Queues statisch angelegt. Das heißt, dass sowohl Namen als auch die damit verknüpften Queueparameter fest hinterlegt sind, um nur ein Beispiel zu nennen. Besser wäre es, alle Parameter in der Eingabedatei des Templates anzugeben. Aus IBM MQ Sicht ist hinzuzufügen, dass die fehlende automatisierte Bereitstellung eines Queue Managers den gewünschten Effekt, einer weitgehende Automatisierung und Unabhängigkeit von der Administration, abschwächt. Während der Realisierung stellte sich ebenfalls heraus, dass ein Template, das mehrere Subsysteme beinhaltet und dadurch sehr anwendungsspezifisch ist, nicht für einen firmenweiten Einsatz geeignet ist. So ist zu empfehlen, dass für jedes Subsystem, also CICS, Db2 und IBM MQ, ein separates Template realisiert wird.

Angenommen es besteht für jedes Subsystem ein Template und das IBM MQ Template beinhaltet die Provisionierung eines Queue Managers, so könnte jeder Entwickler seine eigenen Instanzen der Templates besitzen und beispielsweise für eigene Tests nutzen. Dennoch wäre der ermöglichte Bereitstellungsprozess nicht optimal. So müsste für jede kleine Änderung an der Konfigurationsdatei ein neues Template erzeugt werden, siehe zweiter Fall im Abschnitt 5.4.2. Das dort genannte Beispiel einer CICS-Instanz und eindeutigen Application IDs wird hier aufgegriffen. Eine Möglichkeit dieses Problem zu lösen, wäre einen Pool mit verfügbaren Application IDs bereitzustellen und dann mittels eines Programms eine ungenutzte Application ID zu bestimmen. Dieses Programm kann dann als Step in das Template aufgenommen werden. Jedoch müsste immer noch für jede kleine Änderung an der Konfigurationsdatei ein neues Template erzeugt werden.

Hier schafft z/OSPT Abhilfe. Damit kann, wie in Absatz 2.5.2 beschrieben, mit Hilfe einer Konfigurationsdatei das Template von außen gesteuert werden. Dadurch fällt das Kopieren

des Template für den Mitarbeiter weg, dieser muss mittels des Kommandozeileninterfaces ein Image bauen und daraus einen Container erzeugen. Das Kommandozeileninterface hat einen weiteren Vorteil. Mit dessen Hilfe können Arbeitsschritte für die Provisionierung der Middleware in einen Jenkins-Ablauf aufgenommen werden. Somit läuft der Prozess automatisiert ab und nähert sich modernen Entwicklungsabläufen wie denen aus der Cloud Native Entwicklung an.

Angenommen es existieren jeweils ein CICS, ein Db2 und ein IBM MQ Template und diese sind so realisiert, dass sie firmenweit eingesetzt werden können. Dann wäre der nächste Schritt, die Aufnahme in den „DATEV Marktplatz“, möglich. Der „DATEV Marktplatz“ ist eine Weboberfläche mit der sich Entwicklerteams ihre benötigte PaaS-Umgebung konfigurieren können. Heute stehen ihnen dort Dienste wie MongoDB, PostgreSQL, Kafka und viele weitere zur Verfügung. In weiter Zukunft könnten hier auch Dienste wie CICS, Db2 und IBM MQ zur Auswahl stehen. Dabei ist in Betracht zu ziehen, ob für den User nur bestimmte vorgefertigte Profile, wie „klein“, „mittel“ und „groß“, auswählbar sind. Die im Hintergrund verbundenen Templates und Images müssten dahingehend angepasst werden. Um einen solchen „Service Broker“ zu verwirklichen könnte die von z/OSMF zur Verfügung gestellte REST-API verwendet werden. Diese ermöglicht den Zugriff auf fast alle z/OSMF Funktionalitäten mittels Http-Requests. Für die „Tenant“ Zuweisung zu einem Template wird weiterhin die z/OSMF Oberfläche benötigt. Daran ist zu erkennen, dass von Seiten von z/OSMF beziehungsweise von IBM ebenfalls noch Verbesserungsmöglichkeiten bestehen.

Diese technische Umsetzung ermöglicht in Zukunft den in Diagramm 6.1 dargestellten Bereitstellungsprozess. Es ist zu erkennen, dass Verantwortung von den Administratorenteams an die Entwicklerteams übertragen wird. Dadurch wird Kommunikationsaufwand eingespart und einem Entwickler steht binnen weniger Minuten eine funktionsfähige Laufzeitumgebung für seine legacy z/OS Anwendung zur Verfügung. Bei Problemen oder Beratungswunsch unterstützen die Administratorenteams weiterhin. Für die Realisierung dieser Lösung ist viel Zeitaufwand vor allem auf Seiten der Administration einzuplanen.

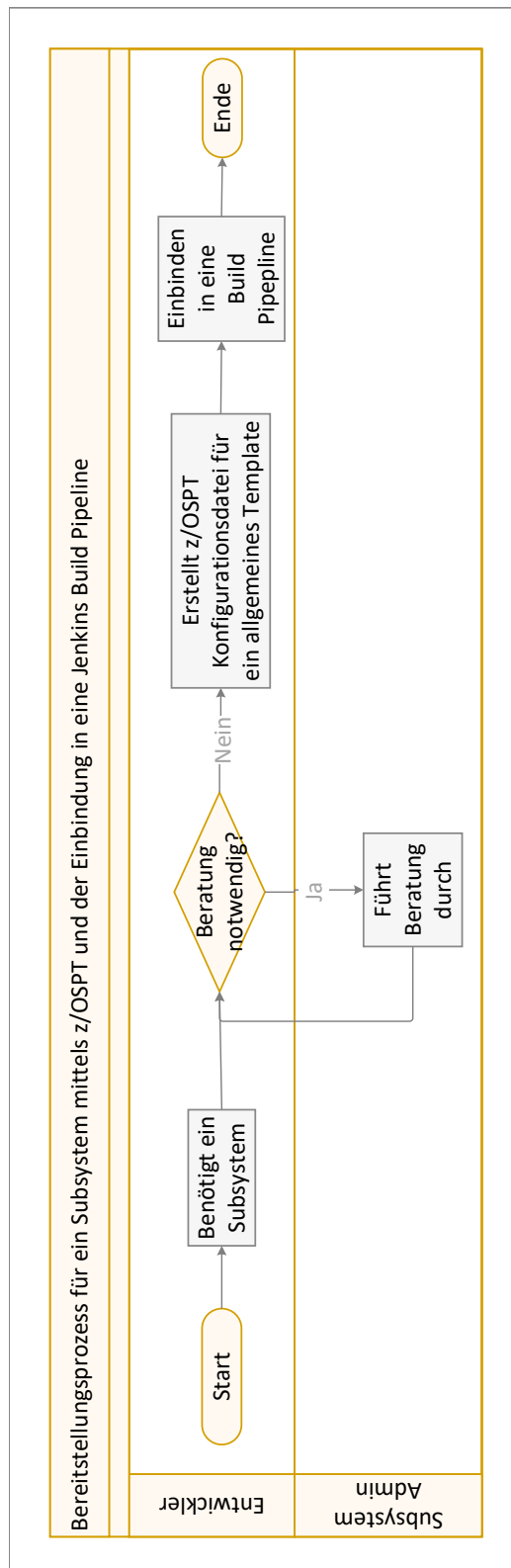


Abbildung 6.1.: Bereitstellungsprozess eines Subsystems mittels einer z/OSPT Konfigurationsdatei

Kapitel 7.

Zusammenfassung

Zusammenfassend lässt sich sagen, dass es generell möglich ist mit dem „IBM Cloud Provisioning and Management for z/OS“-Toolkit Laufzeitumgebungen für legacy z/OS Anwendungen automatisiert bereitzustellen. Das funktionsfähige Template verkürzt den Bereitstellungsprozess deutlich. Durch den Abbau der Kommunikation zwischen den Abteilungen und nur einmaligem Erstellen der Skripte ist es zudem weniger fehleranfällig. Die Stakeholder sehen in diesem Template auch einen Mehrwert. Jedoch ist es noch nicht perfekt. Der Bereitstellungsprozess ist noch immer mit einigen manuellen Schritten verbunden. So muss das Template manuell kopiert werden und Änderungen an der Konfiguration müssen innerhalb des Templates stattfinden.

Hierfür wurde in der Arbeit eine Lösung mit Hilfe von z/OSPT beleuchtet. Diese sieht in einer Endausbaustufe eine einfache Einbindung des Templates in einen automatisierten Build-Prozess, zum Beispiel mit Jenkins, vor. Außerdem würde der Einsatz von z/OSPT das Einbinden in den DATEV eG internen „Marktplatz“ für Cloud Lösungen ermöglichen. Um diese Ziele zu erreichen muss noch viel Aufwand in die Gestaltung des Templates gesteckt werden. Zusätzlich müsste ein sogenannter „Service Broker“ für die Einbindung der einzelnen Subsysteme in den „Marktplatz“ implementiert werden. Diese beiden Lösungsansätze stoßen sowohl bei den Administratorenteams als auch beim involvierten Entwicklerteam auf fruchtbaren Boden. Dadurch wird eine Ähnlichkeit zum Cloud Native-Bereitstellungsprozesses hergestellt. Dies ist ein weiterer Schritt um dem Image eines veralteten Systems mit veraltetem langsamen Prozesses zu entkommen.

Anhang A.

Anhang

A.1. Agenda der neunzehnten Academic Mainframe

Consortium e.V. Tagung vom 16.01.2020 bis 17.01.2020

IBM-Tag am Donnerstag, 16.01.2020

Die Sprecher werden erst im Januar festgelegt

10:00 – 10:20 Uhr

Begrüßung

Wolfram Greis, AMC
Yvette A LaMar, Director, IBM Z Influencer Ecosystem
Roland Trauner, IBM System Z Academic Initiative, Europe

10.20 – 11:00 Uhr

IBM Z15 News / Update

Roland Trauner
IBM System Z Academic Initiative, Europe

11.00 – 12:30 Uhr

Linux Container on IBM Z and LinuxONE

Wilhelm Mild
IBM Executive IT Architect - Integration Architectures for Mobile, IBM Z and Linux
Yulia Gaponenko, Software Developer

12:30 – 13:30 Uhr

Mittagspause

13:30 – 14:30 Uhr

Containers for zOS, Applications and Container Orchestration * Kubernetes / OpenShift

Wilhelm Mild, IBM

14:30 – 15:00 Uhr

Middleware Provisionierung mit zOSMF - eine Bachelorarbeit

David Krug, DATEV eG

15:00 – 15:15 Uhr

Pause

15:15 – 16:00 Uhr

HyperProtect Update

Stefan Liesche
IBM Distinguished Engineer - IBM Hyper Protect Services
Stefan Schmitt
STSM Hyper Protect Services

16:00 – 16:30 Uhr

Offene Punkte, Feedback, weitere Planung

NN, IBM & Wolfram Greis, AMC

16:30 – 17:30 Uhr

History@IBM oder Chiptest Lab Fläche (wahlweise)

für alle Interessierten

Ab 18:00

Netzwerken im IBM Clubheim

für alle Interessierten

Agenda für die Tagung des Academic Mainframe Consortium e.V.

am 16./17.01.2020

AMC-Tagung am Freitag, 17.01.2020

10:00 – 10:15 Uhr

Begrüßung und Vorstellungsrunde

Wolfram Greis, AMC

10:15 – 10:45 Uhr

IBM System Z Academic Initiative 2020

Roland Trauner, IBM

10:45 – 16:00 Uhr

News vom Academic Mainframe Consortium

Wolfram Greis, AMC

Berichte und Diskussionen zu den Arbeitsgruppen

Arbeitsgruppenleiter

Verschiedenes

Alle

Weiteres Vorgehen / Nächstes Treffen

Wolfram Greis, AMC

Die Reihenfolge der Punkte ist noch nicht endgültig festgelegt

A.2. IT workload distribution worldwide in 2018 and 2020, by cloud type

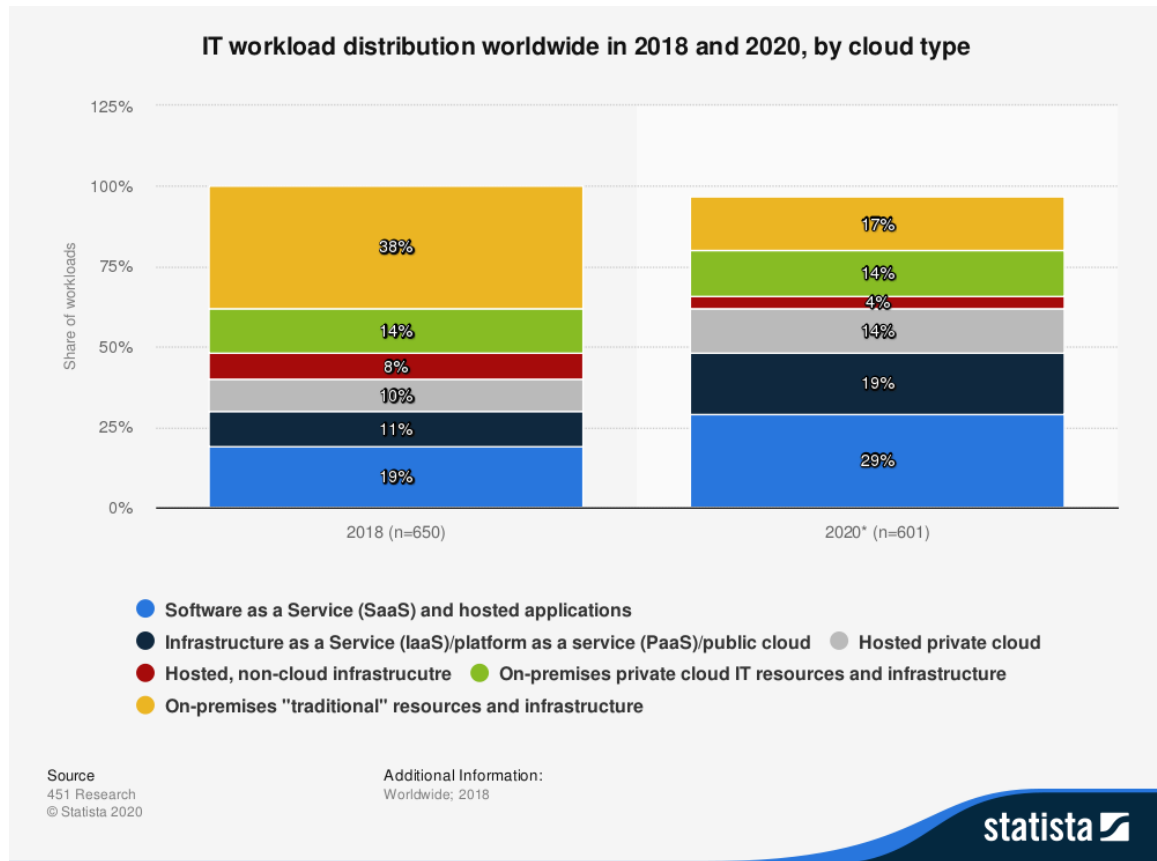


Abbildung A.1.: Weltweiter It Workload im Jahr 2018 und als Vorhersage im Jahr 2020 bei Cloudtyp

A.3. Produktstammdaten Tabellen Data Definition Language

```

1 CREATE TABLE PSSSCHEMA.TPAUSPSSBASELBART
2     (PARTITIONID INTEGER
3                                     NOT NULL
4 WITH DEFAULT 1
5     ,PID INTEGER
6                                     NOT NULL
7     ,AUSPRAEGUNG SMALLINT
8                                     NOT NULL
9     ,GUELTIGAB DATE

```

10		NOT NULL
11	,LFDNR INTEGER	
12		NOT NULL
13	WITH DEFAULT	
14	,GUELTIGBIS DATE	
15		NOT NULL
16	WITH DEFAULT "9999-12-31 "	
17	,PSSID INTEGER	
18	WITH DEFAULT NULL	
19	,ZUSATZID CHARACTER (4) FOR SBCS DATA	
20	WITH DEFAULT NULL	
21	,BEZEICHNUNG VARCHAR (100) FOR SBCS DATA	
22	WITH DEFAULT NULL	
23	,MWSTANTEILFREI DECIMAL (5 , 2)	
24	WITH DEFAULT NULL	
25	,MWSTANTEILREDUZIERT DECIMAL (5 , 2)	
26	WITH DEFAULT NULL	
27	,MWSTANTEILVOLL DECIMAL (5 , 2)	
28	WITH DEFAULT NULL	
29	, CONSTRAINT PID PRIMARY KEY	
30	(PARTITIONID	
31	,PID	
32	,AUSPRAEGUNG	
33	,GUELTIGAB	
34	,LFDNR	
35)	
36)	
37	IN DATABASE PSSBAPRV	
38	APPEND NO	
39	NOT VOLATILE CARDINALITY	
40	DATA CAPTURE NONE	
41	AUDIT NONE	
42	CCSID EBCDIC	
43	PARTITION BY RANGE	
44	(PARTITIONID NULLS LAST ASC	
45)	
46	(PARTITION 1	
47	ENDING (1	
48) INCLUSIVE	
49	, PARTITION 2	

```

50         ENDING ( 2
51     ) INCLUSIVE
52 );
53
54 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSBASELBART
55     ON PSSSCHEMA.TPAUSPSSBASELBART
56     (PARTITIONID ASC
57     ,PID ASC
58     ,AUSPRAEGUNG ASC
59     ,GUELTIGAB ASC
60     ,LFDNR ASC
61     )
62     INCLUDE NULL KEYS
63     CLUSTER
64     PARTITIONED
65     DEFINE YES
66     COMPRESS NO
67     BUFFERPOOL BP2
68     CLOSE YES
69     DEFER NO
70     COPY NO
71     PARTITION BY RANGE
72     (PARTITION 1
73         USING STOGROUP STAPSA01
74             PRIQTY -1
75             SECQTY -1
76             ERASE NO
77         FREEPAGE 0
78         PCTFREE 10
79         GBPCACHE CHANGED
80     ,PARTITION 2
81         USING STOGROUP STAPSA01
82             PRIQTY -1
83             SECQTY -1
84             ERASE NO
85         FREEPAGE 0
86         PCTFREE 10
87         GBPCACHE CHANGED);
88
89 CREATE TABLE PSSSCHEMA.TMAXNUM

```

```

90      (MAXNUMID INTEGER
91
92      ,MAXNUM INTEGER
93
94      ,MAXNUBEZ CHARACTER(42) FOR SBCS DATA
95
96      WITH DEFAULT "X"
97      ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
98
99      WITH DEFAULT "X"
100     ,CONSTRAINT MAXNUMID PRIMARY KEY
101     (MAXNUMID
102     )
103     )
104     IN DATABASE PSSBAPRV
105     APPEND NO
106     NOT VOLATILE CARDINALITY
107     DATA CAPTURE NONE
108     AUDIT NONE
109     CCSID EBCDIC;
110
111     COMMENT ON TABLE PSSSCHEMA.TMAXNUM
112         IS "maximale_□Nummer";
113
114
115     SET CURRENT SQLID = "DB2SADM";
116
117
118     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUMID
119         IS "ID_□fuer_□maximalen_□Nummer";
120
121
122     SET CURRENT SQLID = "DB2SADM";
123
124
125     COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUM
126         IS "maximale_□Nummer";
127
128
129     SET CURRENT SQLID = "DB2SADM";

```

```

130
131
132 COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.MAXNUBEZ
133 IS "Bezeichnung_fuer_maximale_Nummer";
134
135
136 SET CURRENT SQLID = "DB2SADM";
137
138
139 COMMENT ON COLUMN PSSSCHEMA.TMAXNUM.BEZEICHNUNG
140 IS "Bezeichnung_fuer_maximale_Nummer";
141
142 CREATE UNIQUE INDEX PSSSCHEMA.PMAXNUM
143 ON PSSSCHEMA.TMAXNUM
144 (MAXNUMID ASC
145 )
146 INCLUDE NULL KEYS
147 CLUSTER
148 DEFINE YES
149 COMPRESS NO
150 BUFFERPOOL BP2
151 CLOSE YES
152 DEFER NO
153 COPY NO
154 USING STOGROUP STALDL01
155 PRIQTY -1
156 SECQTY -1
157 ERASE NO
158 FREEPAGE 0
159 PCTFREE 99
160 GBPCACHE CHANGED
161 PIECESIZE 2097152K;
162
163 CREATE FUNCTION PSS.WHICH_PARTITIONID
164 (
165 MAXID INTEGER )
166 RETURNS INTEGER
167 VERSION V1
168 DISALLOW DEBUG MODE
169 ASUTIME NO LIMIT

```



```

170  INHERIT SPECIAL REGISTERS
171  WLM ENVIRONMENT FOR DEBUG MODE DB0TWLM
172  APPLICATION ENCODING SCHEME EBCDIC
173  QUALIFIER UGPSENT
174  DYNAMICRULES RUN
175  WITH EXPLAIN
176  WITHOUT IMMEDIATE WRITE
177  ISOLATION LEVEL UR
178  OPTHINT " "
179  REOPT NONE
180  VALIDATE RUN
181  ROUNDING DEC_ROUND_HALF_EVEN
182  DATE FORMAT ISO
183  DECIMAL( 31 )
184  FOR UPDATE CLAUSE REQUIRED
185  TIME FORMAT ISO
186  CURRENT DATA NO
187  DEGREE 1
188  PACKAGE OWNER UGPSENT
189  BUSINESS_TIME SENSITIVE NO
190  SYSTEM_TIME SENSITIVE NO
191  ARCHIVE SENSITIVE NO
192  APPLCOMPAT V10R1
193  LANGUAGE SQL
194  NO EXTERNAL ACTION
195  PARAMETER CCSID EBCDIC
196  DETERMINISTIC
197      NOT SECURED
198      CALLED ON NULL INPUT
199      READS SQL DATA
200      SPECIFIC WHICH_PARTITIONID
201 BEGIN
202     DECLARE MAXNUM INTEGER;
203     SELECT MAXNUM
204         INTO MAXNUM
205         FROM AVADMIN.AMAXNUM
206         WHERE MAXNUMID = MAXID;
207     RETURN MAXNUM;
208 END;
209

```

```

210 SET PATH = "PSS" , "SYSIBM" , "SYSFUN" , "SYSPROC" , "SYSIBMADM" , "PSSSCHEMA" ;
211
212 CREATE VIEW PSSSCHEMA.VPAUSPSS_BASELBART
213     ( PARTITIONID
214       , PID
215       , AUSPRAEGUNG
216       , GUELTIGAB
217       , LFDNR
218       , GUELTIGBIS
219       , PSSID
220       , ZUSATZID
221       , BEZEICHNUNG
222       , MWSTANTEILFREI
223       , MWSTANTEILREDUZIERT
224       , MWSTANTEILVOLL
225     ) AS
226 SELECT B.* FROM TPAUSPSSBASELBART B WHERE B.PARTITIONID =
227     PSS.WHICH_PARTITIONID ( 3011 )
228 ;
229
230 CREATE TABLE PSSSCHEMA.TPAUSPSSPREISE
231     (PARTITIONID INTEGER
232                                     NOT NULL
233 WITH DEFAULT 1
234     ,ARTNR INTEGER
235                                     NOT NULL
236     ,PREISTYPID SMALLINT
237                                     NOT NULL
238     ,GUELTIGAB DATE
239                                     NOT NULL
240     ,STAFFELNR INTEGER
241                                     NOT NULL
242     ,PID INTEGER
243                                     NOT NULL
244 WITH DEFAULT
245     ,PREISREGEL CHARACTER(2) FOR SBCS DATA
246                                     NOT NULL
247 WITH DEFAULT "X"
248     ,GUELTIGBIS DATE
249                                     NOT NULL

```

```

250 WITH DEFAULT "9999-12-31 "
251      ,PRODUKTPREIS DECIMAL(11 , 3)
252 WITH DEFAULT NULL
253      ,EINZELPREIS DECIMAL(8 , 3)
254 WITH DEFAULT NULL
255      ,PREISINTERVALL DECIMAL(11 , 3)
256 WITH DEFAULT NULL
257      ,PREISEINHEIT DECIMAL(8 , 3)
258 WITH DEFAULT NULL
259      ,STAFFELVERTEILUNG CHARACTER(1) FOR SBCS DATA
260 WITH DEFAULT NULL
261      ,INTERVALLVON INTEGER
262 WITH DEFAULT NULL
263      ,INTERVALLBIS INTEGER
264 WITH DEFAULT NULL
265      ,PREISTYPBEZ VARCHAR(100) FOR SBCS DATA
266                                     NOT NULL
267 WITH DEFAULT "X"
268      ,PREISAB DECIMAL(8 , 3)
269 WITH DEFAULT NULL
270      ,PREISABRELEVANZ CHARACTER(1) FOR SBCS DATA
271                                     NOT NULL
272 WITH DEFAULT "K"
273      ,EINHEIT INTEGER
274 WITH DEFAULT NULL
275      ,CONSTRAINT PPAUSPSSPREISE PRIMARY KEY
276      (PARTITIONID
277      ,ARTNR
278      ,PREISTYPID
279      ,GUELTIGAB
280      ,STAFFELNR
281      )
282      )
283      IN DATABASE PSSBAPRV
284      APPEND NO
285      NOT VOLATILE CARDINALITY
286      DATA CAPTURE NONE
287      AUDIT NONE
288      CCSID EBCDIC;
289

```

```

290 CREATE INDEX PSSSCHEMA.IPAUSPSSPREISE
291 ON PSSSCHEMA.TPAUSPSSPREISE
292 (PARTITIONID ASC
293 ,PID ASC
294 )
295 INCLUDE NULL KEYS
296 NOT CLUSTER
297 DEFINE YES
298 COMPRESS NO
299 BUFFERPOOL BP2
300 CLOSE YES
301 DEFER NO
302 COPY NO
303 USING STOGROUP STAPSA01
304 PRIQTY -1
305 SECQTY -1
306 ERASE NO
307 FREEPAGE 0
308 PCTFREE 10
309 GBPCACHE CHANGED
310 PIECESIZE 2097152K;
311
312 CREATE INDEX PSSSCHEMA.IPAUSPSSPREISE2
313 ON PSSSCHEMA.TPAUSPSSPREISE
314 (PARTITIONID ASC
315 ,ARTNR ASC
316 ,PREISTYPID ASC
317 ,GUELTIGAB ASC
318 ,INTERVALLVON ASC
319 )
320 INCLUDE NULL KEYS
321 NOT CLUSTER
322 DEFINE YES
323 COMPRESS NO
324 BUFFERPOOL BP2
325 CLOSE YES
326 DEFER NO
327 COPY NO
328 USING STOGROUP STAPSA01
329 PRIQTY -1

```

```

330          SECQTY -1
331          ERASE NO
332      FREEPAGE 0
333      PCTFREE 10
334      GBPCACHE CHANGED
335      PIECESIZE 2097152K;
336
337 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSPREISE
338      ON PSSSCHEMA.TPAUSPSSPREISE
339      (PARTITIONID ASC
340      ,ARTNR ASC
341      ,PREISTYPID ASC
342      ,GUELTIGAB ASC
343      ,STAFFELNR ASC
344      )
345      INCLUDE NULL KEYS
346      CLUSTER
347      DEFINE YES
348      COMPRESS NO
349      BUFFERPOOL BP2
350      CLOSE YES
351      DEFER NO
352      COPY NO
353      USING STOGROUP STAPSA01
354          PRIQTY -1
355          SECQTY -1
356          ERASE NO
357      FREEPAGE 0
358      PCTFREE 10
359      GBPCACHE CHANGED
360      PIECESIZE 2097152K;
361
362 CREATE TABLE PSSSCHEMA.TPAUSPSSBART
363      (PARTITIONID INTEGER
364
365      NOT NULL
366      WITH DEFAULT 1
367      ,PID INTEGER
368
369      NOT NULL
370      ,ARTNR INTEGER
371
372      NOT NULL

```

370	WITH DEFAULT	
371	,ANDAT DATE	
372		NOT NULL
373	WITH DEFAULT "1966-02-14"	
374	,OPDATBEN CHARACTER (1) FOR SBCS DATA	
375		NOT NULL
376	WITH DEFAULT "J"	
377	,AUSDIENSTREL CHARACTER (1) FOR SBCS DATA	
378		NOT NULL
379	WITH DEFAULT "N"	
380	,VERTRIEBSREL CHARACTER (1) FOR SBCS DATA	
381		NOT NULL
382	WITH DEFAULT "N"	
383	,VERTRELDAT DATE	
384	WITH DEFAULT NULL	
385	,KOMMASTELLEN INTEGER	
386	WITH DEFAULT NULL	
387	,ERTRNR INTEGER	
388		NOT NULL
389	WITH DEFAULT	
390	,GFEDNR INTEGER	
391		NOT NULL
392	WITH DEFAULT	
393	,UPLONR INTEGER	
394		NOT NULL
395	WITH DEFAULT	
396	,MWSTEUERSATZ INTEGER	
397	WITH DEFAULT NULL	
398	,POLINR INTEGER	
399		NOT NULL
400	WITH DEFAULT	
401	,EXPGNR INTEGER	
402		NOT NULL
403	WITH DEFAULT	
404	,EXPONR INTEGER	
405		NOT NULL
406	WITH DEFAULT	
407	,ARTIKELTYPID INTEGER	
408		NOT NULL
409	WITH DEFAULT	

410	,ARTIKELTYPALT CHARACTER (4) FOR SBCS DATA	
411		NOT NULL
412	WITH DEFAULT "0000 "	
413	,BERBESTEINHID INTEGER	
414		NOT NULL
415	WITH DEFAULT	
416	,BERBESTEINHALT CHARACTER (4) FOR SBCS DATA	
417		NOT NULL
418	WITH DEFAULT "0000 "	
419	,LEISTGRUPID INTEGER	
420		NOT NULL
421	WITH DEFAULT	
422	,LEISTGRUPALT CHARACTER (4) FOR SBCS DATA	
423		NOT NULL
424	WITH DEFAULT "0000 "	
425	,BERFREQID INTEGER	
426		NOT NULL
427	WITH DEFAULT	
428	,NUTZERID INTEGER	
429	WITH DEFAULT	
430	,LEISTARTID INTEGER	
431		NOT NULL
432	WITH DEFAULT	
433	,BERFREQALT CHARACTER (4) FOR SBCS DATA	
434		NOT NULL
435	WITH DEFAULT "0000 "	
436	,LEISTARTALT CHARACTER (4) FOR SBCS DATA	
437		NOT NULL
438	WITH DEFAULT "99 "	
439	,NUTZERALT CHARACTER (1) FOR SBCS DATA	
440		NOT NULL
441	WITH DEFAULT "K"	
442	,BARTBEZ_20 CHARACTER (20) FOR SBCS DATA	
443		NOT NULL
444	WITH DEFAULT "X"	
445	,ARTIKELTYPBEZ CHARACTER (50) FOR SBCS DATA	
446		NOT NULL
447	WITH DEFAULT "Keine□Zuordnung"	
448	,BERBESTEINHBEZ CHARACTER (50) FOR SBCS DATA	
449		NOT NULL

450	WITH DEFAULT "Keine□Zuordnung"	
451	,LEISTGRUPBEZ CHARACTER (50) FOR SBCS DATA	
452		NOT NULL
453	WITH DEFAULT "Keine□Zuordnung"	
454	,BERFREQBEZ CHARACTER (50) FOR SBCS DATA	
455		NOT NULL
456	WITH DEFAULT "Keine□Zuordnung"	
457	,NUTZERBEZ CHARACTER (50) FOR SBCS DATA	
458		NOT NULL
459	WITH DEFAULT "Keine□Zuordnung"	
460	,LEISTARTBEZ CHARACTER (50) FOR SBCS DATA	
461		NOT NULL
462	WITH DEFAULT "Keine□Zuordnung"	
463	,BARTBEZ_100 VARCHAR (100) FOR SBCS DATA	
464		NOT NULL
465	WITH DEFAULT "X"	
466	,ERTRBEZ VARCHAR (100) FOR SBCS DATA	
467		NOT NULL
468	WITH DEFAULT "X"	
469	,GFEDBEZ VARCHAR (100) FOR SBCS DATA	
470		NOT NULL
471	WITH DEFAULT "X"	
472	,UPLOBEZ VARCHAR (100) FOR SBCS DATA	
473		NOT NULL
474	WITH DEFAULT "X"	
475	,POLIBEZ VARCHAR (100) FOR SBCS DATA	
476		NOT NULL
477	WITH DEFAULT "X"	
478	,EXPGBEZ VARCHAR (100) FOR SBCS DATA	
479		NOT NULL
480	WITH DEFAULT "X"	
481	,EXPOBEZ VARCHAR (100) FOR SBCS DATA	
482		NOT NULL
483	WITH DEFAULT "X"	
484	,HAKONR INTEGER	
485	WITH DEFAULT NULL	
486	,HAKOBEZ VARCHAR (100) FOR SBCS DATA	
487	WITH DEFAULT NULL	
488	,INPGNR INTEGER	
489		NOT NULL


```

490 WITH DEFAULT
491     ,INPGBEZ VARCHAR(100) FOR SBCS DATA
492                                     NOT NULL
493 WITH DEFAULT "X"
494     ,BEZ035 CHARACTER(35) FOR SBCS DATA
495                                     NOT NULL
496 WITH DEFAULT "X"
497     ,CONSTRAINT PPAUSPSSBART PRIMARY KEY
498     (PARTITIONID
499     ,PID
500     )
501     ,CONSTRAINT UPAUSPSSBART UNIQUE
502     (PARTITIONID
503     ,ARTNR
504     )
505     )
506     IN DATABASE PSSBAPRV
507 APPEND NO
508 NOT VOLATILE CARDINALITY
509 DATA CAPTURE NONE
510 AUDIT NONE
511 CCSID EBCDIC
512 PARTITION BY RANGE
513     (PARTITIONID NULLS LAST ASC
514     )
515     ( PARTITION 1
516         ENDING ( 1
517         ) INCLUSIVE
518         , PARTITION 2
519         ENDING ( 2
520         ) INCLUSIVE
521         );
522
523 CREATE UNIQUE INDEX PSSSCHEMA.PPAUSPSSBART
524     ON PSSSCHEMA.TPAUSPSSBART
525     (PARTITIONID ASC
526     ,PID ASC
527     )
528     INCLUDE NULL KEYS
529     CLUSTER

```

```

530 PARTITIONED
531 DEFINE YES
532 COMPRESS NO
533 BUFFERPOOL BP2
534 CLOSE YES
535 DEFER NO
536 COPY NO
537 PARTITION BY RANGE
538 (PARTITION 1
539         USING STOGROUP STAPSA01
540             PRIQTY -1
541             SECQTY -1
542             ERASE NO
543         FREEPAGE 0
544         PCTFREE 10
545         GBPCACHE CHANGED
546 ,PARTITION 2
547         USING STOGROUP STAPSA01
548             PRIQTY -1
549             SECQTY -1
550             ERASE NO
551         FREEPAGE 0
552         PCTFREE 10
553         GBPCACHE CHANGED);
554
555 CREATE UNIQUE INDEX PSSSCHEMA.UPAUSPSSBART
556     ON PSSSCHEMA.TPAUSPSSBART
557     (PARTITIONID ASC
558     ,ARTNR ASC
559     )
560     INCLUDE NULL KEYS
561     NOT CLUSTER
562     DEFINE YES
563     COMPRESS NO
564     BUFFERPOOL BP2
565     CLOSE YES
566     DEFER NO
567     COPY NO
568     USING STOGROUP STAPSA01
569         PRIQTY -1

```

```
570          SECQTY -1
571          ERASE NO
572      FREEPAGE 0
573      PCTFREE 10
574      GBPCACHE CHANGED
575      PIECESIZE 2097152K;
576
577 CREATE UNIQUE INDEX PSSSCHEMA.UPAUSPSSBART
578      ON PSSSCHEMA.TPAUSPSSBART
579      (PARTITIONID ASC
580      ,ARTNR ASC
581      )
582      INCLUDE NULL KEYS
583      NOT CLUSTER
584      DEFINE YES
585      COMPRESS NO
586      BUFFERPOOL BP2
587      CLOSE YES
588      DEFER NO
589      COPY NO
590      USING STOGROUP STAPSA01
591          PRIQTY -1
592          SECQTY -1
593          ERASE NO
594      FREEPAGE 0
595      PCTFREE 10
596      GBPCACHE CHANGED
597      PIECESIZE 2097152K;
```

A.4. Interview Fragebögen

Vorlage

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

7. Können Sie sich vorstellen, mit dem Toolkit täglich zu arbeiten?

8. Wenn 7. Mit ja beantwortet wurde, begründen Sie ihre Meinung.

9. Wenn 7. Mit nein beantwortet wurde, was müsste sich ändern, dass dem so wäre?

10. Freitext für sonstiges und Anmerkungen:

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

+ flexibel Versionierung und Publish

- Startschwierigkeiten viele verschiedene Sprachen und Dokumentarten

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

+ APIfizierung

+ Container-Gedanke

+ konfigurierbar über PT-File von außerhalb der Templates

-Template muss sehr dynamisch sein

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

+ lediglich editieren der Files

-Fehlendes Highlighting für „mixed JCL“

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

- Hoher Ersteinrichtungsaufwand

- Einarbeitung

- Abschreckende Wirkung (verschiedene Sprachen usw.)

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Das Template ist aus CICS-Sicht ablauffähig und mehrfach einsetzbar.

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Großer Vorteil ist die Flexibilität durch den Einsatz von Variablen. Nachteil ist die damit verbundene Komplexität des dahinterliegenden Templates.

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
Die Oberfläche wurde vom CICS-Team bisher nicht genutzt, daher ist keine Bewertung möglich. Zudem ist wenig Erfahrung mit vergleichbaren Tools wie Cloud Foundry vorhanden.
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Keine Toolunterstützung. Editieren per Notepad ohne Highlighting. Keine sprechenden Fehlermeldungen. Dokumentation der JCL-Skriptsprache nicht vorhanden.

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)
Prinzipiell kann mit dem Toolkit alles erreicht werden, allerdings ist sehr viel Anpassungsarbeit notwendig, um es auf die Firmengegebenheiten anzupassen. Hilfreich wären mehr Beispiele, bessere Dokumentation, Step by Step Anleitung oder ein Wizard.
6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?
Hoher manueller Aufwand zu erbringen. Kein SelfService für den Entwickler vorhanden. Vorherige Abstimmung zwischen Sysprog und Entwicklung notwendig.
7. Können Sie sich vorstellen, mit dem Toolkit täglich zu arbeiten?
Jein, man könnte es sich vorstellen, aber es gibt auch viele Hürden.

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Der Ablauf zeigt, dass eine Automatisierung möglich ist. Diese Erkenntnis ist sehr wertvoll. Um die Provisionierung aber wirklich nutzen zu können ist noch viel Weiterentwicklung notwendig.

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Für den aktuellen Stand der Untersuchung halte ich z/OSPT noch nicht für relevant. In der Endausbaustufe (Automatisiertes Deployment innerhalb einer CI/DC-Pipeline z.B. mit Jenkins) wird ein CLI-Interface wie z/OSPT aber sehr wichtig und vereinfacht die Nutzung für Entwickler

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

(wenn es ohne automatisches Deployment gemacht wird)

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

Mächtiges Tool aber auch sehr komplex

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

Er funktioniert sehr gut aber man ist von anderen Personen abhängig und hat dadurch natürlich Wartezeiten, die man mit einem automatischen Prozess eliminieren würde.

7. Können Sie sich vorstellen, mit dem Toolkit täglich zu arbeiten?

Ja

8. Wenn 7. Mit ja beantwortet wurde, begründen Sie ihre Meinung.

Nur wenn man es kapselt (jenkins)

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Ablauf ist für mich nachvollziehbar. Siehe auch Anmerkungen (Frage 10)

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Die Nutzung von Z/OSPT macht Sinn, denn die Commands (z.B. build und run) sind meiner Meinung nach einfach in einen Quellcode zu integrieren und geläufig. Problematisch sehe ich jedoch die Verwendung der Begriffe Container und Image, da hier Begriffe vertauscht und synonym verwendet werden.

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Besser wäre es, wenn zur Bearbeitung eine IDE verwendet werden würde (Syntax)

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

Zukunft des modernen Deployments auf dem Mainframe. Ähnlich der offenen Welt. Bringt die Plattform z nach vorne!

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

Aktuell noch sehr komplex. Die Bereitstellung ist aktuell noch sehr anwendungsspezifisch und sehr statisch. Es wird ein sehr umfangreiches Wissen über alles beteiligten Subsysteme benötigt. Hoher Konfigurationsaufwand und Vorarbeit von Nöten (Rechtekonzept, Funktionsuser, etc.)

7. Können Sie sich vorstellen, mit dem Toolkit täglich zu arbeiten?

Ja.

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- Mehrwert dadurch, dass mehr Verantwortung bei dem Entwickler ist
- Weniger händische Eingriffe

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>

- Wird das von IBM noch weiterentwickelt?
- Features, die bereits vorhanden sind, sind schon gut
- Flexibilität ist höher als mit dem Ablauf aus 1.

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Da noch nicht selbst damit gearbeitet wurde, kann es nicht beurteilt werden.

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

- Gezeigtes ist gut, aber Zeitaufwand ist mit einzubeziehen und die zu leistenden Vorarbeiten
- MQ Queue Manager ist komplexer → noch mehr Zeitaufwand und notwendige Vorarbeiten werden mehr

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

- Deutlich manueller Arbeitsablauf
- Viele Rückfragen und viel Arbeiten auf Zuruf, Kommunikation über Email, Telefon oder Termine

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- Fehlender Umfang (speziell für MQ)
- Nur spezifische Queues mit speziellen Parametern

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- Weiterhin Abstimmung mit Dritten (RACF, IP, Storage) notwendig
- Nur „pseudo“ Docker Container

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- Automation nicht vorhanden
 - o Zum Beispiel keine Analogie zur Jenkins-Replay-Funktion

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

- Viele gute Ansätze
- Nicht einfach genug zu verwenden im Vergleich zu Jenkins und einer PaaS Lösung

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

- Deutlich manueller Arbeitsablauf
- Viele Rückfragen und viel Arbeiten auf Zuruf, Kommunikation über Email, Telefon oder Termine

7. Können Sie sich vorstellen, mit dem Toolkit täglich zu arbeiten?

Ja und Nein

Entwickler 1

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- Es wird viel Wissen (bezüglich der Oberfläche und des Templates) benötigt und dieses muss auch bei geringer Nutzung über einen längeren Zeitraum erhalten werden.
- Außerdem ist weiterhin viel Zuarbeit der Administration notwendig.
- Es stellt sich die Frage, wer die DDL für Datenbanken erstellt.
- Ansonsten schon ganz gut.

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

- Vor allem ein Ausblick für eine Nutzung in der QS und Produktionsstages
- Ablauf mehr in den Händen des eigenen Teams → höhere Effizienz, aber auch höhere Verantwortung.
- Unkomplizierte Nutzung mittels Jenkins und den „DATEV Marktplatz“

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

- Zeitaufwändig durch viele Absprachen bezüglich Erstaufwand.
- Wenns läuft, läuft.
- Abhängigkeit von dritten (z.B. Admins)

7. Können Sie sich vorstellen, mit dem Toolkit täglich zu arbeiten?

Ja

Mitarbeiter des Technologiestrategieteams

1. Es wurde ein Template für die Rechnungsschreibung, welches ein CICS, die benötigten MQ Queues und theoretisch die benötigte Db2 Datenbanken beinhaltet, vorgestellt. Der Ablauf, der damit einhergeht, beschränkt sich zunächst auf z/OSMF. Bewerten Sie diesen, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

z/OSMF stellt sich für mich als eine Oberfläche dar, die ich für eine solche „Task“ nutzen kann. Sieht einfach aus. Besser würde es mir gefallen, wenn ich den bereits existierenden „Marketplace“ unserer DATEV Cloud Lösung nutzen könnte.

2. Es wurde eine Ergänzung mit z/OSPT, zu oben genannten Ablauf, erläutert. Bewerten Sie diese, begründen Sie Ihre Bewertung.

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>

Unkomplizierte Nutzung mittels Jenkins und den „DATEV Marktplatz“ Die Konfiguration eines Skriptes mit z/OSPT ist für die Einbindung der Provisionierung in automatische Build-Prozesse hilfreich, und damit wichtig. Es muss sich jemand um den Aufbau der Build-Pipeline kümmern, (Build Engineer) die Rolle haben wir aktuell in den z/OS Projekten noch nicht. Aber es macht Sinn und bringt die z/OS Anwendungen näher an die Vorgehensweise der Cloud Native Entwicklung.

3. Bewerten Sie folgende Punkte bezüglich der Benutzerfreundlichkeit der Oberfläche:
- a. Verwaltung der Templates in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

- b. Verwaltung der Instanzen in z/OSMF (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

4. Bewerten Sie die gezeigte Arbeitsweise für Änderungen an den Workflow Definitionsdateien. (nicht für Entwickler relevant)

1	2	3	4	5
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

5. Wie ist Ihr erster persönlicher Eindruck zum Toolkit? (nicht für Entwickler relevant)

6. Wie würden Sie den aktuellen Bereitstellungsprozess beurteilen?

Aktuell teilen sich n Entwickler an einem Produkt die gleiche Entwicklungsumgebung, arbeiten im gleichen CICS und auf der gleichen Test-Datenbank. Änderungen beeinflussen auch die Tests der anderen Kollegen, und müssen koordiniert werden. Die

A.5. Workflow Step mit REST-Call

”

```

1 <step name="db2_create_database">
2     <title>db2_create_database.</title>
3     <description>Erzeugt mit der Hilfe des DB2
4     Service Brokers eine Datenbank</description>
5     <instructions substitution="false">
6     Erzeugt DB2 Datenbank.</instructions>
7     <weight>10</weight>
8     <skills>REST</skills>
9     <autoEnable>true</autoEnable>
10    <rest>
11    <httpMethod>PUT</httpMethod>
12    <schemeName substitution="false">http</schemeName>
13    <hostname substitution="false">hostname</hostname>
14    <uriPath>uriPath</uriPath>
15    <requestBody substitution="true">
16    {
17        "service_id": "${instance-DFH_DB2_SERVICEID}" ,
18        "plan_id": "${instance-DFH_DB2_PLANID}" ,
19        "organization_guid": "DUMMY" ,
20        "space_guid": "DUMMY" ,
21        "parameters": {
22            "GROUP": "UGZTAL" ,
23            "VUSERID": "${_step-stepOwnerUpper}" ,
24            "DBNAME": "${instance-DFH_DB2_DATABASENAME}" ,
25            "UserID": "${_step-stepOwnerUpper}" ,
26            "Passwort": "DUMMY"
27        }
28    }
29    </requestBody>
30    <expectedStatusCode>201</expectedStatusCode>
31    <requestHeaders substitution="false">
32    { "Authorization": "Basic VDMwMTkzQTpYZjN1I2RJNA==" }
33    </requestHeaders>
34    </rest>
35 </step>

```

Abbildungsverzeichnis

1.1. Anteil der verwendeten Programmiersprachen auf dem Mainframe bei DATEV eG in Prozent	3
1.2. Auszug aus einem REXX Skript in der ISPF Oberfläche	4
2.1. Abgrenzung von Continuous Integration, Continuous Delivery und Continuous Deployment (Quelle: [http 20b])	9
2.2. Continuous Integration Prozessaufbau (Quelle: [http 20c])	10
2.3. Architekturübersicht über die Subsysteme einer Stage bei DATEV eG	12
2.4. Etablierte Konfigurationsoberfläche am Beispiel bei Änderung einer Transaktion	15
2.5. z/OSMF Willkomens Ansicht	18
2.6. z/OSPT mögliche Kommandozeilenbefehle	20
4.1. Bereitstellungsprozess einer CICS Instanz	28
4.2. Bereitstellungsprozess einer Db2 Datenbank	30
4.3. Bereitstellungsprozess einer IBM MQ Queue	32
5.1. Login Bildschirm der provisionierten DATEV spezifischen CICS-Instanz	45
5.2. Define IBM Queue, am Beispiel einer Trigger Queue	46
5.3. Beispiel einer Fehlermeldung von zOSMF	53
6.1. Bereitstellungsprozess eines Subsystems mittels einer z/OSPT Konfigurationsdatei	61
A.1. Weltweiter It Workload im Jahr 2018 und als Vorhersage im Jahr 2020 bei Cloud-typ	68

Tabellenverzeichnis

5.1. Vergleich zwischen z/OSPT und z/OSMF	38
5.2. Zu verändernde Variablen im „cics_getting_started“-Template	39
5.3. Zu verändernde Variablen im „cics_54“-Template	40
5.4. Vergleich der beiden Templates im Bezug auf deren Umfang	41

Quellcodeverzeichnis

5.1. Hinzufügen weiterer CSD Gruppen zur Liste der provisionierten CICS-Instanz mittels eines Jobs	42
5.2. Setzen der SIT Parameter durch Auslesen der „DFH_REGION_SITPARAMS“ Variablen	44
5.3. Erstellung einer neuen CSD Gruppe	50
5.4. Auslesen der „DFH_MQ_QUEUEENAMES“ Variablen und schreiben in REXX Variablen	53
5.5. Zur Laufzeit erzeugtes Skript, der Grundlage aus Codeabschnitt 5.4	54
listings/ddl.txt	68
listings/db2provision.xml	94

Literaturverzeichnis

- [Also 93] S. Alsop. “IBM still has the brains to be a player in client/server platforms”. *InfoWorld*, Vol. 15, No. 10, p. 4, 1993.
- [Aran 13] C. Aranha. *IBM WebSphere MQ V7.1 and V7.5 features and enhancements*. IBM redbooks, IBM Corp. International Technical Support Organization, Poughkeepsie, NY, 1st ed. Ed., 2013.
- [Cass 07] P. Cassier. *System programmer’s guide to Workload manager*. IBM redbooks, IBM International Technical Support Organization, United States?, 4th ed. Ed., 2007.
- [Ceru 03] P. E. Ceruzzi. *A history of modern computing*. *History of computing*, MIT Press, Cambridge, Mass., 2. ed. Ed., 2003.
- [Ebbe 11] M. Ebbers, J. Kettner, W. O’Brien, and B. Ogden. *Introduction to the new mainframe: Z/OS basics*. IBM redbooks, IBM Corporation International Technical Support Organization, Poughkeepsie, NY, third edition , (march 2011) Ed., 2011.
- [http 19a] “<https://searchengineland.com/google-now-handles-2-999-trillion-searches-per-year-250247>”. 02.12.2019.
- [http 19b] “<https://www.datev.de/web/de/m/ueber-datev/das-unternehmen/geschichte/>”. 25.11.2019.
- [http 19c] “<https://www.ibm.com/it-infrastructure/z/cics>”. 23.11.2019.
- [http 20a] “<https://amc-ev.org/>”. 23.2.2020.
- [http 20b] “<https://blog.infotelcorp.com/blog/articles/the-mainframe-skills-gap-is-widening-automation-can-save-us>”. 25.2.2020.
- [http 20c] “<https://medium.com/jorgeacetozi/continuous-integration-vs-continuous-delivery-vs-continuous-deployment-d5839a85a959>”. 25.2.2020.
- [http 20d] “<https://neuhandeln.de/im-ueberblick-die-groessten-anbieter-fuer-cloud-computing/>”. 27.2.2020.
- [http 20e] “<https://pepgotesting.com/continuous-integration/>”. 25.2.2020.

- [http 20f] “<https://www.cloudcomputing-insider.de/was-ist-cloud-foundry-a-615766/>”. 23.2.2020.
- [http 20g] “<https://www.cloudcomputing-insider.de/was-ist-cloud-native-a-669681/>”. 23.2.2020.
- [http 20h] “<https://www.datev.de/web/de/m/ueber-datev/das-unternehmen/kurzprofil/>”. 27.2.2020.
- [http 20i] “<https://www.egovernment-computing.de/was-ist-ein-legacy-system-a-802283/>”. 22.2.2020.
- [http 20j] “<https://www.fintechfutures.com/2018/11/ibm-shows-off-three-new-bank-clients/>”. 25.2.2020.
- [http 20k] “<https://www.hochschulkompass.de/home.html>”. 09.02.2020.
- [http 20l] “https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.izua700/izupr”. 26.2.2020.
- [http 20m] “https://www.ibm.com/support/knowledgecenter/SSXH44_1.1.0/zospt/cics/zospt-cics-properties.html”. 26.2.2020.
- [Kim 14] G. Kim. *The Phoenix project: A novel about IT, DevOps, and helping your business win*. IT Revolution Press, Portland, Oregon, 2014.
- [Last 17] B. Laster. *Continuous Integration vs. Continuous Delivery vs. Continuous Deployment*. O’Reilly Media, Inc, 1st edition Ed., 2017.
- [Love 13] M. Lovelace. *VSAM demystified. IBM redbooks*, IBM Corp. International Technical Support Organization, Poughkeepsie, NY, 3rd ed. Ed., 2013.
- [Rayn 11] C. Rayns. *CICS transaction server from start to finish. Redbooks*, IBM International Technical Support Organization, Poughkeepsie, N.Y., 2011.
- [Rott 18] R. J. T. Rotthove. *IBM z/OS Management Facility V2R3. Redbooks*, IBM Redbooks, [Place of publication not identified], 2018.
- [Stee 03] B. Steegmans. *DB2 for z/OS and OS/390: Ready for Java. IBM redbooks*, IBM, International Technical Support Organization, [S.l.], 1st ed. Ed., 2003.
- [Vohr 16] D. Vohra. *Pro Docker. The expert’s voice in open source*, Apress, Berkeley, 2016.