



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: Krug

Vorname: David

Matrikel-Nr.: 3036355

Fakultät: Informatik

Studiengang: Informatik

Semester: Wintersemester

2019/2020

Titel der Abschlussarbeit:

Automatisierte Provisionierungsmechanismen für Laufzeitumgebungen von Legacy z/OS Anwendungen mit "IBM Cloud Provisioning and Management for z/OS" am Beispiel der "Rechnungsschreibung" bei DATEV eG

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 2 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

Sperrvermerk

Die vorliegende Arbeit beinhaltet interne vertrauliche Informationen der DATEV e.G. Sie ist nur für die Beteiligten an der Begutachtung bestimmt. Die Weitergabe des Inhalts der Arbeit im Gesamten oder in Teilen sowie das Anfertigen von Kopien oder Abschriften – auch in digitaler Form – vor dem Ablauf der Sperrfrist von 2 Jahren untersagt. Ausnahmen bedürfen der schriftlichen Genehmigung der DATEV e.G.

Kurzdarstellung

Ziel dieser Arbeit ist es, zu bestimmen, ob die Bereitstellung von Laufzeitumgebungen für legacy z/OS Anwendungen über einen cloud nativen, Platform-as-a-Service Ansatz bei DATEV e.G. möglich ist. Es werden folgende Forschungsfragen gestellt:

- 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?

Dafür wurde anhand einer Beispielanwendung von der DATEV e.G. das Tool „IBM Cloud Provisioning and Management for z/OS“ untersucht. Es wurden zwei vorhandene Möglichkeiten aufgezeigt, eine davon implementiert. Für ein Meinungsbild bezüglich Mehrwertes und Akzeptanz des Tools, wurden Interviews mit Stakeholdern durchgeführt. Diese Bild zeigt, dass in dem Tool eine Chance auf Verbesserung der aktuellen Prozesse gesehen wird.

Ergebnis war auch, dass die implementierte Variante nicht optimal für den Praxiseinsatz bei DATEV e.G. ist, aber eine wichtige Basis für die automatisierte Bereitstellung von Laufzeitumgebungen für z/OS Anwendungen darstellt. Weiterführende Forschung könnte darauf aufbauend Variante zwei untersuchen und Möglichkeiten einer weiteren, praxisgeeigneteren Optimierung des z/OS Bereitstellungsprozesses aufzeigen.

Vorwort

Die vorliegende Bachelorarbeit entstand im Rahmen meines Verbundstudiums bei der DATEV e.G. in Nürnberg.

Für die Betreuung meiner Bachelorarbeit möchte ich mich bei Prof. Dr.-Ing. Korbinian Riedhammer und Prof. Dr. rer. nat. Friedhelm Stappert bedanken.

Für ihre tatkräftige Unterstützung möchte ich auch meiner Betreuerin in der Firma DATEV e.G., Sabine Lauterbach, herzlich danken.

Außerdem danke ich allen Kollegen aus dem CICS-, Db2-, IBM MQ-Administratorenteams und dem Entwicklerteam der DATEV-Rechnungsschreibung dafür, dass ich jeder Zeit mit Fragen und Anliegen auf sie zugehen durfte und mir immer freundlich weitergeholfen wurde.

Nicht zuletzt danke ich meiner Familie und Freunde für die Motivation und den Beistand während meines Bildungsweges, welcher mir dadurch deutlich erleichtert wurde.

Inhaltsverzeichnis

1. Einleitung	3
1.1. Problemstellung	5
1.2. Ziel der Arbeit	8
2. Grundlagen	9
2.1. cloud native bei DATEV e.G.	9
2.1.1. „Cloud Foundry“	10
2.1.2. „CI/CD-Pipeline“	10
2.2. Mainframe / Großrechner	12
2.3. Mainframe Anwendungen bei DATEV e.G.	13
2.4. Subsysteme / Middleware	13
2.4.1. Customer Information Control System	14
2.4.2. Db2	15
2.4.3. IBM MQ	16
2.5. „IBM Cloud Provisioning and Management for z/OS“	17
2.5.1. z/OS Management Facility	20
2.5.2. z/OS Provisioning Toolkit	21
3. Vorgehensweise	25
4. Analyse	29
4.1. Analyse von cloud native bei DATEV e.G.	29
4.2. Aktueller Bereitstellungsprozess	30
4.2.1. Bereitstellung einer CICS Instanz	31
4.2.2. Bereitstellungsprozess einer Db2 Datenbank	33
4.2.3. Bereitstellungsprozess einer IBM MQ Queue	35
4.2.4. Zusammenfassung aktueller Bereitstellungsprozess	35
4.3. DATEV-Rechnungsschreibung	38
4.3.1. Tägliche Bewertung	39
4.3.2. Preisermittlung	39
5. Realisierung	41
5.1. Vergleich zwischen z/OSPT und z/OSMF	41

5.2. Testplex	42
5.2.1. „cics_getting_started“-Template	43
5.2.2. „cics_54“-Template	44
5.3. Entwicklungsstages	50
5.3.1. CICS Anpassung	51
5.3.2. Db2 Anpassung	52
5.3.3. IBM MQ Anpassung	53
5.3.4. Testablauf	54
5.4. Bereitstellungsprozess aktuelles Template	56
5.4.1. Use-Case: Neue Template Instanz	56
5.4.2. Use-Case: Zusätzliche Template Instanz	56
5.4.3. Use-Case: Änderungen durch Administratorenteam	57
5.5. Fazit Realisierung	57
5.6. Interviews	61
5.6.1. CICS Administratoren	61
5.6.2. Db2 Administratoren	61
5.6.3. Meinungsbild	63
6. Ausblick	65
7. Zusammenfassung und Fazit	69
A. Anhang	71
A.1. Verwendete Versionen der z/OS Komponenten im Rahmen dieser Arbeit	71
A.2. Agenda der neunzehnten Academic Mainframe Consortium e.V. Tagung vom 16.01.2020 bis 17.01.2020	71
A.3. IT workload distribution worldwide in 2018 and 2020, by cloud type	74
A.4. Fragen an die IBM	74
A.5. Interview Fragebögen	82
A.6. Workflow Step mit REST-Call	92
A.7. Produktstammdaten Tabellen Data Definition Language	93
Abbildungsverzeichnis	109
Tabellenverzeichnis	111
Quellcodeverzeichnis	113
Literaturverzeichnis	115
Glossar	117

Begriffe, die im Glossar erläutert werden, werden bei ihrem ersten Auftreten **rot** markiert.

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 Kernabläufe einen IBM Mainframe. Dies beinhaltet 87 Prozent aller Kreditkartentransaktionen und ca. 350.000 Transaktionen pro Sekunde.⁹ Inklusive dieser Transaktionen verarbeiten

¹[Also 93, S. 4]

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, S. 6]

⁷[Ceru 03, Kap. 5]

⁸[http 20j]

⁹[http 20k]

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, IBM 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. So werden pro Tag circa 150.000 **Batch Jobs** und circa 90 Millionen CICS-Transaktionen verarbeitet. Diese Last wird von circa 14.000 aktiven Modulen erzeugt. Wie in der Abbildung

¹⁰^[http 19c]

¹¹^[http 19a]

¹²^[http 20l]

¹³^[http 20e]

¹⁴^[http 20k]



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

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.¹⁵

1.1. Problemstellung

Im Jahre 2020 ist der größte Konkurrent für den Mainframe die Cloud. Laut einer Vorhersage aus dem Jahr 2018¹⁶ 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 DevOps¹⁷ Prozesse aufgebaut. Dies ist im Cloud-Zeitalter nötig, um mit einer verbesserten Entwicklungseffizienz und neuen Architekturen Anwendungen („Apps“) schneller auf den Markt bringen und auf Kundenanforderungen schneller reagieren zu können. Stichwort: Continuous Integration, Continuous Delivery (CI/CD)¹⁸. Ein Baustein der effizientern Prozesse durch PaaS sind die sog. „Self Services“. D.h., Entwicklerteams können sich über sog. „Cloud Services“, z.B. Datenbanken wie PostgreSQL, Mongo und Messaginglösungen wie Kafka entweder manuell über einen Marktplatz (siehe Abbildung

¹⁵[\[http 20i\]](http://20i)

¹⁶Statistik im Anhang A.1

¹⁷Siehe Absatz 4.1

¹⁸Siehe Absatz 2.1.2

1.2) oder automatisiert per „Build-Pipeline“ eine Laufzeitumgebung für ihre Anwendung zusammenbauen. Eine genaue Beschreibung der Begrifflichkeiten erfolgt im Absatz 2.1. Den

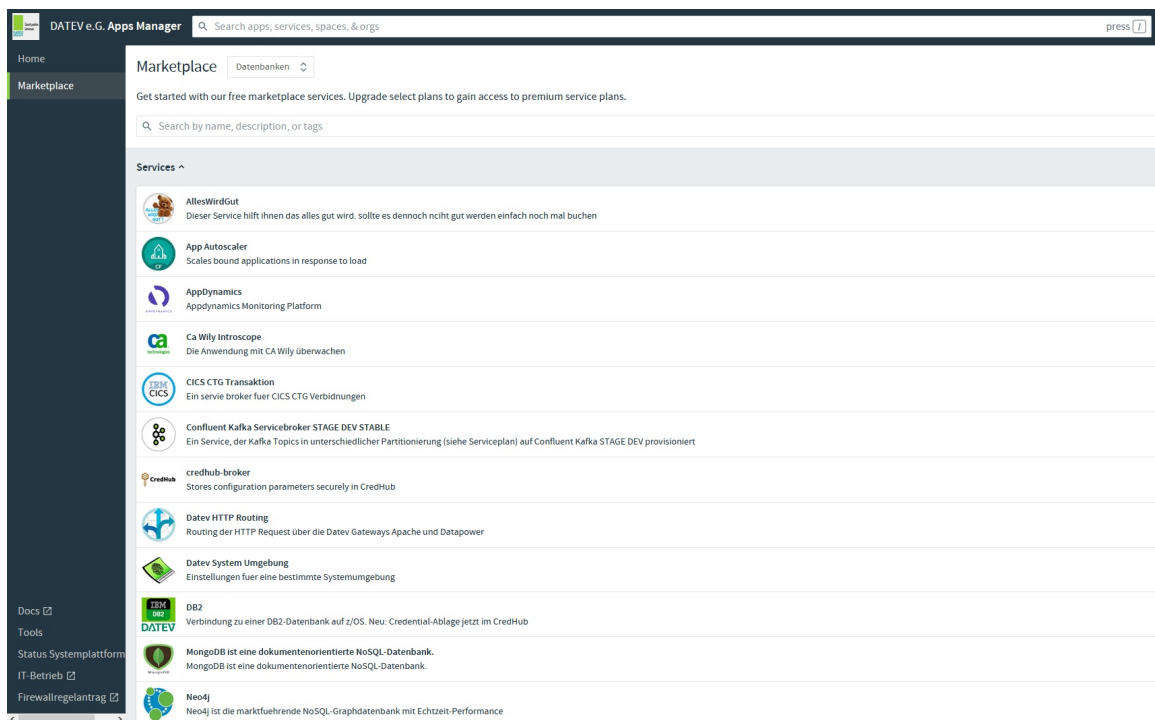


Abbildung 1.2.: Cloud Foundry Marketplace bei DATEV e.G.

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 1.3 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 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 moder-

¹⁹Integrated Development Environment wie IntelliJ oder Eclipse


```

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  /* -----+*/
000062  Generate:
000063  /* -----+*/
000064  /* Template abfragen -----+*/
000065  /* -----+*/
000066  rce = inq_template()
000067  if rce > 0 then return rce
000068
000069  Do l=1 to Loop
000070
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    if template_qtype = 'QREMOTE' then
000077      command = "DEFINE REPLACE QR("q_new") LIKE("q_template")"
000078      "QSGDISP("template_qsgdisp") RNAME("q_new")"
000079    /* -----+*/
000080    /* alle 10 Schleifen kurz warten -----+*/
000081    /* -----+*/
000082    if l//10 = 0 then
000083      Call Wait 1
000084    rce = command_send()
000085    if rce > 0 then leave
000086
000087  End
000088  return rce
000089  /* -----+*/

```

Abbildung 1.3.: Auszug aus einem REXX Skript in der ISPF Oberfläche

nisiert, 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 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 Bestandsgegeschäfts, 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.

²⁰??

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

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, „weil man so etwas bisher nicht vermisst hat“²². IBM bietet eine Lösung mit dem „IBM Cloud Provisioning and Management for z/OS“-Toolkit. Dies hat sich noch nicht flächendeckend durchgesetzt, aber „Viele Kunden haben [...] im Moment Interesse, jedoch warten viele hier auf die ersten Erfahrungen von anderen“²³. 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 Administratoren-teams?

Um diese Fragen zu beantworten wird die Provisionierung einer z/OS Laufzeitumgebung für eine bestehende Anwendung untersucht. Diese Anwendung sollte CICS als Anwendungsserver, 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.

²²Marcel Amrein, IBM Senior Technical Sales Professional (MQ and CICS) (Quelle: Anhang A.4)

²³Tobias Leicher, IBM Senior IT Specialist for CICS and zAPI (Quelle: Anhang A.4)

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. 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-Computing** 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 **Docker**. Die Docker Container werden über Images beschrieben und beinhalten neben der Anwendung alles das, was die Anwendung an Komponenten zur Laufzeit benötigt, Bibliotheken, Dateien.¹ Somit können Anwendungen auf verschiedenen „privat“ und „public“ Cloud-Umgebungen, auch von unterschiedlichen Anbietern, ausgeführt werden. Große Anbieter sind hier beispielsweise AWS (Amazon), Google, Microsoft Azure.² Dort können bereitstehende Services für Datenhaltung, Security, Messaging usw. genutzt werden. [[http 20h](#)]

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

- „Cloud Foundry“
- CI/CD-Pipeline

¹[[Vohr 16](#), Kap. 1]

²[[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.³ 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 beige-steuert. 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. Über die Programmiersprache groovy können diese Kommandozeilenbefehle in eine Jenkins basierende CI/CD-Pipeline aufgenommen werden. So kann die Bereitstellung über mehrere Stages hinweg automatisiert werden. Diese Methode wird bei der DATEV e.G. eingesetzt. [[http 20g](#)]

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

³[[http 20m](#)]



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

kann eine erste Überprüfung des Codes mittels statischer Codeanalyse durchgeführt werden. Dazu zählt unter anderem die Prüfung vorher definierter Coderichtlinien. 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, Kap. 2]

„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, Kap. 3]



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, Kap. 4]

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. [?, S. 9-10]

2.3. Mainframe Anwendungen bei DATEV e.G.

Das Betriebssystem des IBM Mainframes ist das für zero downtime stehende z/OS.⁴ 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 Versionswechsel⁵, 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.

⁴[S. 92][?]

⁵circa 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. [?, S. 4]

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. [?, 5-8]

2.4.1.3. CICS Anwendungsdeployment

Zur Beantwortung der Forschungsfragen liegt der Fokus auf der Entwicklungsstage, deshalb wird im Folgenden nur das Deployment von Anwendung in eine Entwicklungs-CICS Instanz dargestellt. Das eine Anwendung einer CICS Instanz zur Verfügung steht, muss der kompilierte Sourcecode der Anwendung manuell in einer der CICS Instanz bekannten Bibliothek abgelegt werden. Eine solche Bibliothek muss im „Started Task Control-Job“⁶ einer CICS Instanz manuell angegeben werden.

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.

⁶Siehe Absatz 4.2.1

Bei DB0T handelt es sich um ein Datenbankmanagementsystem, dass sich alle Anwendungen der Entwicklungsstage teilen. Die darin enthalten Datenbanken werden von der Db2 Administration verwaltet. 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. [?, Kap. 3.2]

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. [?, S. 36]

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. [?, S. 37-38]

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. [?, Kap. 4.5]

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

Die Verwaltung von Subsystemen von z/OS ist eine hochspezialisierte, teilweise manuelle Tätigkeit.⁷ 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.

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“⁸, 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 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

⁷Analyse des aktuellen Bereitstellungsprozesses, siehe in Absatz 4.2

⁸Siehe Absatz 2.5.1

```

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

„AMC“⁹ 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.2 zu finden. Der AMC e.V. ist ein Förderverein für die akademische Ausbildung auf dem Mainframe.¹⁰ 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“¹¹. 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 „Aktiondefinitionfile“ und das

⁹Academic Mainframe Consortium e.V.

¹⁰<http://20a>

¹¹AMC Vorstand Ernst Lugert, 20.1.2020

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 20o](#)]

Ein Workflow ist über eine XML Datei, das sogenannte „Workflowdefinitionfile“, 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 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](#), S. 140]

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

„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 20o](#)]

„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. [[Keit 16](#), S. 15]

„IBM Cloud Provisioning and Management for z/OS“ umfasst zwei Lösungen, „z/OS Management Facility“¹² und „z/OS Provisioning Toolkit“¹³.

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.

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](#)]

¹²siehe Absatz [2.5.1](#)

¹³siehe Absatz [2.5.2](#)

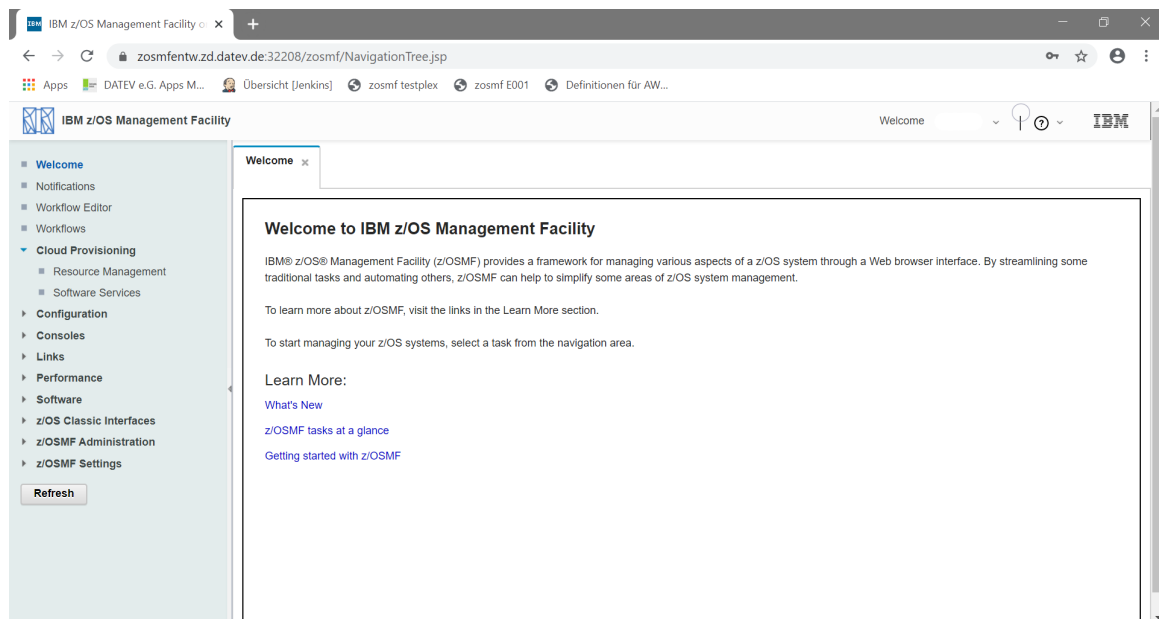


Abbildung 2.5.: z/OSMF Willkommens Ansicht

Unter dem Punkt „Resource Management“ werden die „Domains“ und „Tenants“ verwaltet. Zur Verwaltung der Templates und Template-Instanzen kommen die „Software Services“ 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, S. 8]

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. Genau wie bei Cloud Foundry (Siehe Absatz 2.1.1) ermöglicht dieses die Einbindung in eine Jenkins basierende CI/CD-Pipeline. 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, Kap. 1]

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 vorhandenen 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             [-h (--help)]                                     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>]  Start one or more containers
  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 Entwicklungsphase 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 Entwicklungsstages. In der Entwicklungsstages 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 semistrukturale 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.5 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, dass 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. Mittels des Kommandozeileninterfaces kann die Bereitstellung auch in eine CI/CD-Pipeline integriert werden. 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 eigens dafür zuständige Administratorenteams die Middleware auf allen Stages. 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 Bereiche, an denen das „IBM Cloud Provisioning and Management for z/OS“-Tool helfen kann, identifizieren 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, z.B. um eine neue Fachlichkeit testen zu können. Hier hilft das CICS Administratorenteam mittels Beratung. 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, die dann aber meist wieder zwischen Anwendungen und Entwicklern geteilt wird. 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 des dieser Arbeit zugrunde liegenden Beispiels handelt es sich um siebzehn verschiedene VSAM Dateien. Diese Dateien benötigt die CICS-Instanz um zum Beispiel Systemfehler zu protokollieren oder den Debugger aktivieren zu können.

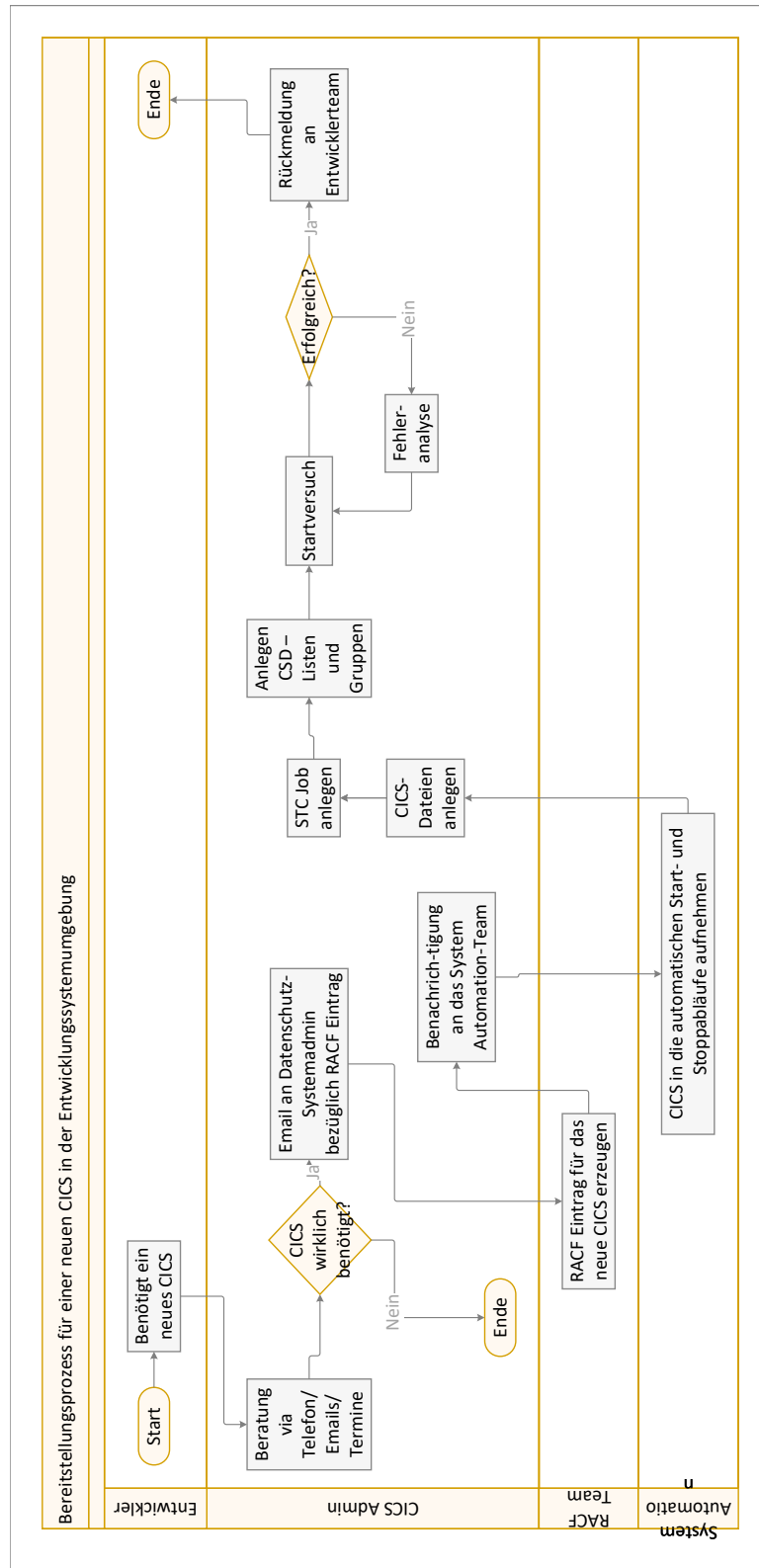


Abbildung 4.1.: Bereitstellungsprozess einer CICS Instanz

„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 Datenbankmanagementsystem und welchem MQ Queue Manager sich diese Instanz verbinden soll. Somit können Transaktionen auf diese beiden Ressourcen zugreifen.

Started Task Control-Job

Bei einem Started Task Control-Job, kurz STC Job, handelt es sich um einen langlaufenden 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 (z.B. CICS) ein solcher Job. In diesem werden zunächst einige zur Laufzeit benötigten Bibliotheken und Dateien eingebunden, unter anderem die CICS spezifischen Dateien⁴. Außerdem werden hier die SIT⁵ 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 dazugehörenden CSD und die Information, ob eine Verbindung zu einem Db2 Datenbankmanagementsystem und/oder zu einem Queue Manager hergestellt werden soll.

Nach der Durchführung dieser Schritte wird das CICS gestartet und es 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 Projektkürzel⁶, der Datenbank- und Projektname und die Projektbezeichnung müssen mit den involvierten Abteilungen besprochen werden. Über den sogenannten „Datenbankänderungsantrag“ wird ein Genehmigungsprozess angestoßen. Wenn

⁴Beschreibung in Absatz 4.2.1

⁵CICS system initialization table

⁶eindeutige Kennung von DATEV Projekten

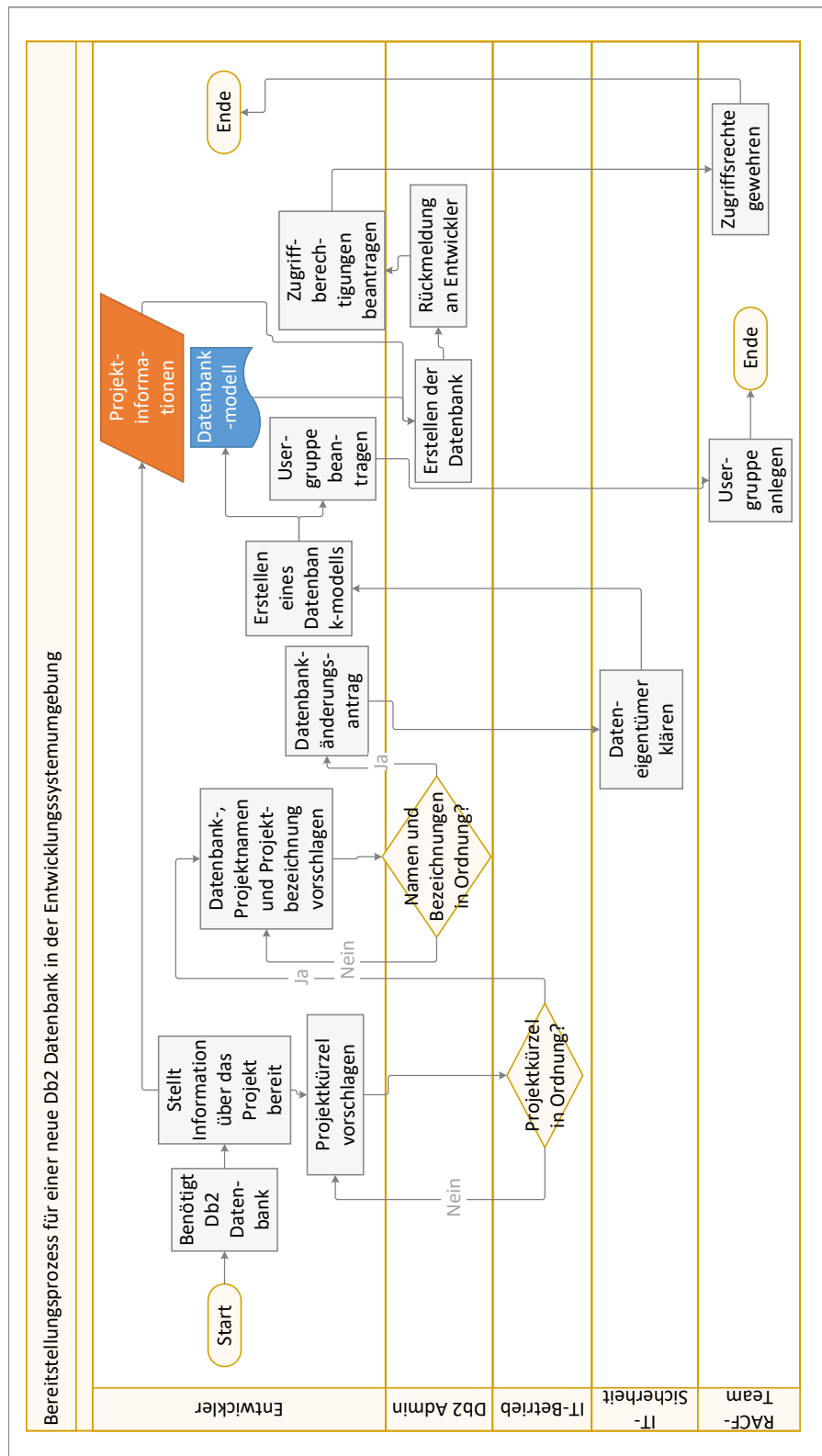


Abbildung 4.2.: Bereitstellungsprozess einer Db2 Datenbank

alle Genehmigungen erteilt wurden, kann ein Dateneigentümer festgelegt werden. Anschließend muss die Datenbank mittels eines Datenbankmodells vom Entwicklerteam beschrieben 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 angeforderte 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, wie im Fallbeispiel dieser Arbeit, der DATEV-Rechnungsschreibung, so muss der entsprechende Eintrag in die CSD Datei manuell aufgenommen werden. 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 muss der Queue Manager auf dem die Queue angelegt wurde dem CICS mittels einem entsprechenden Eintrag in die CSD Datei mitgeteilt werden. 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 mit vielen manuellen Schritten verbunden. Der Hauptaufwand liegt bei den Administratorenteams, die in verschiedenen organisatorischen Einheiten angesiedelt sind. Das Entwicklerteam ist Initiator des Ablaufs. Folglich kümmert es sich um die erforderlichen Formulare und die erste Kontaktaufnahme zum jeweiligen Administratorenteam.

Zusätzlich zu den zahlreichen manuellen Schritten sind die vielen Absprachen zwischen mehreren Abteilungen zu nennen. Steht ein beteiligtes Team nicht zu Verfügung, oder sind andere Projekte höher priorisiert, kommt es zu Verzögerungen, das Entwicklungsteam muss warten, der komplette Zeitplan kann sich dadurch nach hinten verschieben. Der Prozess für

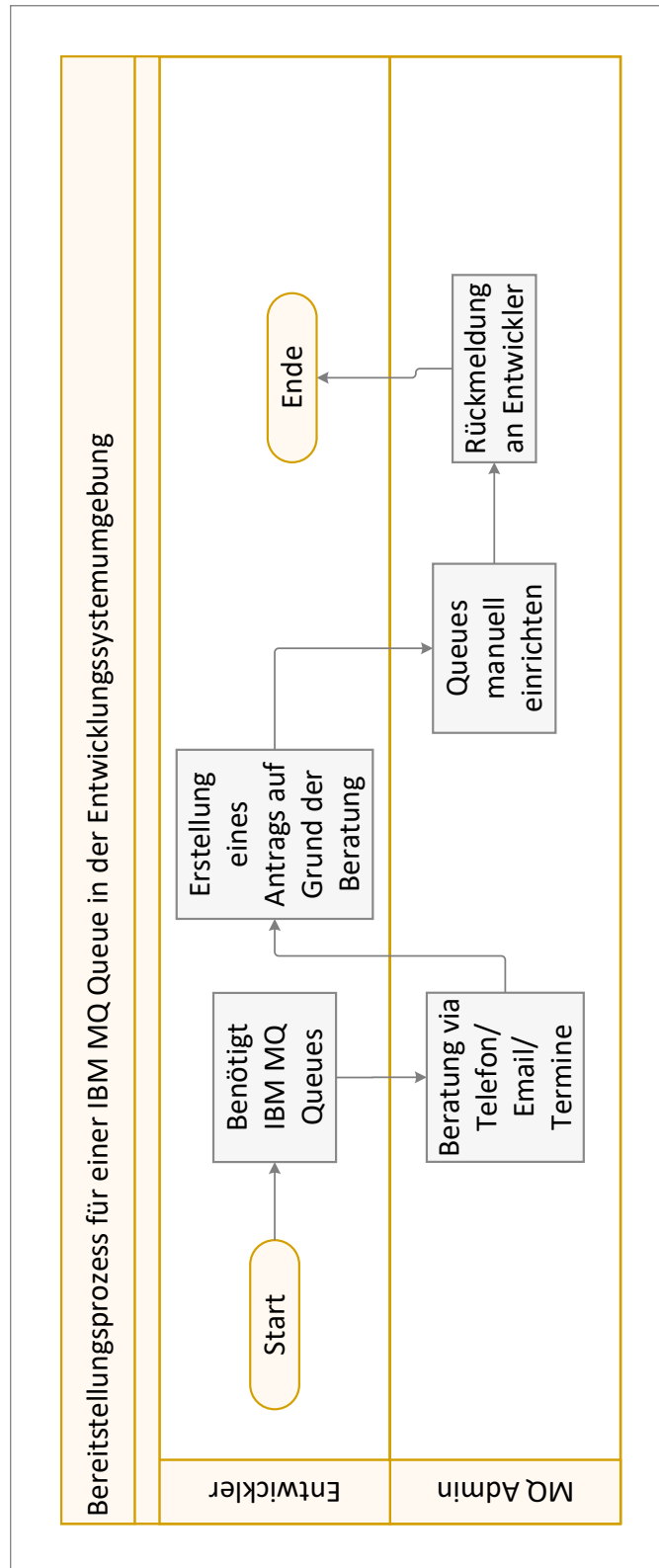


Abbildung 4.3.: Bereitstellungsprozess einer IBM MQ Queue

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

Der Start des gesamten Prozesses findet meist auf „Zuruf“ statt. So existiert für die erste Kontaktaufnahme kein Formular, keine Automation oder ähnliches. Es wird auf E-Mail, Telefon oder mittels Terminen zurückgegriffen.

Wird jetzt der Vergleich zu dem Bereitstellungsprozess im cloud native-Umfeld⁷ in folgenden Punkten gezogen, so steht dem automatisierten und dadurch effizienten cloud native Prozess, ein durch viele Absprachen langsamer und fehleranfälliger Prozess auf z/OS gegenüber.

- Produktautonomie
- Deployment
- Architektur

Produktautonomie

Die Autonomie eines Product-Teams ist hier nicht gegeben, die Administratorenteams sind für den Betrieb der Subsysteme verantwortlich. Daran wird sich auch durch automatisierte Provisionierung in Zukunft nichts ändern, da hier hohes Spezialistenwissen benötigt wird, das nicht von jedem Product-Team bereitgestellt werden kann. Dazu kommt, dass die Provisionierung im Rahmen dieser Arbeit auch nur die Bereitstellung von Entwicklungs-Instanzen der Middleware Systeme betrachtet. Die Administratorenteams sind aber insbesondere für die produktiven Instanzen im Betrieb verantwortlich, die nach wie vor von Anwendungen geteilt werden. D.h. der Cloud-Aspekt des automatischen Deployments einer Anwendung inklusive Runtime zwischen Stages wird also nicht umgesetzt.

Deployment

Da für das Deployment einer z/OS CICS Anwendung, wie in Absatz 2.4.1.3 beschrieben, Bibliotheken zum Einsatz kommen, muss beim Testen einer Anwendung immer geprüft werden, ob der aktuellste Stand der Anwendung in dieser Bibliothek abgelegt ist, ob die Bibliothek überhaupt der CICS Instanz zur Verfügung steht und ob nicht eine andere Bibliothek mit anderem Stand von der CICS Instanz angezogen wird. Diese Probleme verschlechtern sich, wenn an der gleichen Anwendung mehrere Mitarbeiter entwickeln, so ist viel Absprache notwendig, um beispielsweise nicht den Stand eines Kollegen zu überschreiben.

⁷Siehe Absatz 4.1

Architektur

Neben den Problemen im Bereitstellungsprozess ist die grundlegende Architektur einer z/OS Anwendung zu betrachten. Im Vergleich zur „Microservices“-Architektur, die im cloud native Umfeld häufig zum Einsatz kommt⁸, sind klassische z/OS Anwendungen monolithisch aufgebaut. Insbesondere aus Performancegründen, haben sich über die Jahre Abhängigkeiten und Verzahnungen zwischen einzelnen Anwendungen gebildet. Diese starke Koppelung erhöht die Komplexität und erschwert das Testen dieser Anwendungen.

Wird der Architektur Aspekt der geteilten Subsystem Ressourcen betrachtet, kristallisiert sich ein weiteres Problem heraus. Es ist möglich, dass eine Anwendung eine komplette CICS Instanz zum Absturz bringt. Ist dies der Fall sind auf Grund einer Anwendung alle anderen Anwendungen dieser CICS Instanz nicht mehr verfügbar. Mit Hilfe von isolierten Testumgebungen kann dieser Problem gelöst werden.

Nach diesem Vergleich sieht die DATEV e.G. bei dem hier untersuchten automatisierten Prozess vor allem bei der Entwicklungseffizienz einen Vorteil. Dabei wird der Fokus auf die Effizienz bei der Bereitstellung isolierter Testumgebungen, ohne Wartezeit und mit möglichst wenig manueller Interaktion gelegt. Dadurch verspricht sich die DATEV e.G. kürzere Entwicklungszyklen und weniger Fehleranfälligkeit durch zu hohen Abstimmungsaufwand.

Hierzu wird in dieser Arbeit das Tool „IBM Cloud Provisioning and Management for z/OS“ für die automatisierte 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:

- Sammeln von Berechnungssätze
- Tägliche Bewertung

⁸Siehe Absatz 2.1

- 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 wird die Berechnung in CICS durchgeführt und in folgende 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 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

⁹Abrechnungs- und Umsatz-Basis

notwendig, da die notwendige Performance aufgrund von bis zu 60 Millionen Datenbankzugriffen nicht erreicht werden 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

Bei beiden steht sowohl eine schnelle Provisionierung, als auch Deprovisionierung von Instanzen durch den Entwickler im Vordergrund. Mittels Templates läuft beides automatisiert, verlässlich, und ohne manuelle Abstimmungen ab. Die Bereitstellung einer Instanz ist wiederholbar, so kann die Instanz z.B. im Fehlerfall sicher neu erstellt werden und muss nicht langlebig gepflegt werden.

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

Es folgt ein Vergleich der beiden Tools an Hand folgender Kriterien:

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

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 leichter. Hinzu kommt, dass innerhalb von z/OSPT für die Zuweisung von Domains und Tenants eine weitere externe Konfigurationsdatei gepflegt werden muss.

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 RACF 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 Entwicklungsstuge nicht ohne weiteres umsetzbar ist.

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

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.²

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 dem Aktiondefinitionfile getestet. Dabei wurde sich auf das Nutzen der z/OSMF Oberfläche fokussiert und nicht auf die eigentliche Funktionsfähigkeit der CICS-Instanz.

²Siehe 2.5

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 in Tabelle 5.3 folgende Variablen gesetzt werden. Die Kurzbeschreibungen und die Beschreibungen aller weiteren Variablen, die im Standard Template vorhanden sind, ist unter [\[http 20n\]](http://20n) 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 ³ für die Sprachum- gebung ⁴
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 erst einmal 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 auch in einer automatisiert provisionierten CICS-Instanz 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“-Prinzip⁵ 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.

⁵„You aren’t gonna need it“-Prinzip

Die nächste Voraussetzung für die Bereitstellung einer CICS Instanz ist das Erstellen einer CSD-Datei⁶. Folgende Entscheidung wurde in Zusammenarbeit mit dem CICS Administratorenteam getroffen. Um Auswirkungen auf bereits im Einsatz befindliche Instanzen zu 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 gemeinsamen CSD Datei für alle provisionierten Instanzen müssten Änderungen, wie beispielsweise die Verfügbarkeit einer neueren CICS Version, an einer weiteren Stellen angepasst werden. Wird als Basis aber die aktuell schon verwaltete CSD Datei verwendet, genügt eine Neuprovisionierung des Templates um die Änderung auch für die provisionierten Instanzen zu übernehmen. 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 das 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.

⁶Beschreibung siehe Absatz 4.2.1


```

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 Instanz beschrieben. In der Systemumgebung Testplex bedeutet dies die Provisionierung der Datenbank Instanz ohne Tabelle und Daten.

Für die Erstellung einer Db2 Datenbank existiert innerhalb der DATEV e.G. bereits ein „Self Service“. Dieser stellt für die Provisionierung einer Db2 Datenbank auch eine REST-API zur Verfügung. Wie im Absatz 2.5 beschrieben, ist es möglich, innerhalb eines Workflow Steps einen REST-Request abzuschicken. Der Code ist in Abbildung A.6 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 dem DB0C Sandbox Datenbankmanagementsystem⁷ 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, sowie die Aufnahme weiterer Bibliotheken in der „createCICS.jcl“. Dieser Aufruf wurde mittels neuer Variablen im Template möglichst dynamisch gestaltet und mussten in dem Variableinputfile gesetzt werden.

⁷Siehe Absatz 2.4.2

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.

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 Testplex. 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 würden. 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

⁸User ID mit zunächst keinen Berechtigungen

```

***** 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

technischen Usern ausführen zu dürfen. Für Db2 ist ein solcher User nicht notwendig, da es sich beim Datenbankmanagementsystem hinter der REST-API um DB0C⁹ handelt und jeder Entwickler in dieser „Sandbox“ Datenbanken verwalten darf.

Bei der Übertragung des Templates vom Testplex in die Entwicklungsstages waren Anpassungen in allen drei Bereichen des Templates notwendig.

5.3.1. CICS Anpassung

Damit 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 (Kommentar: Link?), die vor jeden Job des Templates eingefügt wird, zu hinterlegen. Die CICS spezifischen Dateien können von der täglichen Datensicherung der Entwicklungsstages ausgeschlossen werden, da diese bei der Deprovisionierung gelöscht werden. Um dies zu gewährleisten, musste der Messageclass Parameter mit dem Wert „NONE“ angegeben werden.

⁹Siehe Absatz 2.4.2

Als Vorlage für die spezifische CSD Datei, die im Template genutzt wird, wird die Standard Entwicklungsstage CSD genutzt. In der Entwicklungsstage kommen im Vergleich zum Testplex andere Db2 und IBM MQ Bibliotheken zum Einsatz. Dahingehend wurde die „createCICS.jcl“-Datei angepasst.¹⁰ Zusätzlich musste ein SIT Parameter angepasst werden, so dass die Log Dateien in der Entwicklungsstage funktionsfähig sind. Eine weitere CSD Gruppe musste hinzugefügt werden. Siehe 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., mit dem es möglich ist, neue Programmversionen, und Schnittstellen zu bereits produktiven Modulen nutzen zu können.

5.3.2. Db2 Anpassung

Um die Datenbankstruktur der DATEV-Rechnungsschreibung im DB0C Datenbankmanagementsystem nach zu stellen, wurde eine genaue technische Analyse der DATEV-Rechnungsschreibungsdatenbank durchgeführt. Dabei stellte sich heraus, dass die Komplexität der benötigten Tabellen sehr hoch ist. 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 Views¹¹. 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.

Dennoch wurde von der Db2 Administration mit der Erstellung der dafür notwendigen DDL Skripte begonnen. Es stellte sich, dass bereits für einen kleinen Teil an Tabellen circa 600 Zeilen DDL Code¹² notwendig ist. Eine vollständige Provisionierung eines so komplexen Datenbanksystems übersteigt den zeitlichen Rahmen dieser Arbeit. Sollte sich die Provisionierung generell als zielführend erweisen wird dieser Einmalaufwand erbracht werden. Im weiteren Lebenszyklus der Anwendung kann auf diese Db2 jederzeit zugegriffen werden, Änderungen sind ein erheblich geringerer Aufwand als die Initialleistung.

Für die weiteren Schritte der hier vorliegenden Arbeit werden deshalb die bereits im Datenbankmanagementsystem DB0T vorhandenen Datenbanken der DATEV-Rechnungsschreibung 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

¹⁰Code siehe Anhang ??

¹¹Alias einer Datenbankabfrage, auf die wie auf eine normale Tabelle zugegriffen werden kann

¹²Im Anhang A.7 zu finden

der Deprovisionierungsdatei die Datenbanksteps auskommentiert, da diese für den weiteren Test nicht mehr zum Einsatz kommen.

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 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 die maximale Queuetiefe und die maximale Länge einer einzelnen Nachricht festgelegt. Im Original-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 Queuetiefe 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-Administrations 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 Rech-

nungsschreibung bereitgestellt. 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, 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 in der provisionierten Laufzeitumgebung zu starten, musste lediglich der verwendete Queue Manager angepasst werden. Über die Queues und das verwendete Triggering wird die Transaktion im provisionierten CICS gestartet. Um das Ergebnis der Anwendung zu prüfen, wurde dies mit dem gleichen Testablauf und den gleichen Eingabedateien in der aktuell existierenden Testumgebung der DATEV e.G. Rechnungsschreibung durchgeführt. Bei einem anschließenden Vergleich der Ausgabedateien beider Läufe waren keine Abweichungen festzustellen.

5.4. Bereitstellungsprozess aktuelles Template

Bei dem Bereitstellungsprozess, der durch das aktuelle Template möglich gemacht wird, sind drei Fälle zu unterscheiden: (Kommentar, das sind die technischen Use-Cases, aber in welchen fachlichen könnten die notwendig sein?)

1. Use-Case: Neue Template Instanz

Ein Entwickler möchte eine Programmänderung auf einer isolierten Laufzeitumgebung testen. Dadurch wird eine neue Instanz eines Templates benötigt. Dem Entwicklerteam steht das dieses Template in z/OSMF zur Verfügung und es wurde noch keine Instanz dieses Templates provisioniert.

2. Use-Case: Zusätzliche Template Instanz

Ein weiterer Entwickler möchte eine andere Programmänderung auf einer isolierten Laufzeitumgebung testen. Dadurch wird eine neue Instanz eines Templates benötigt. Dem Entwicklerteam steht dieses Template in z/OSMF zur Verfügung aber es steht bereits eine Instanz dieses Templates zur Verfügung. Um diese Instanz nicht zu stören, wird eine weitere Instanz benötigt.

3. Use-Case: Änderungen durch Administratorenteam

Die Provisionierung von einer IBM MQ Instanz soll von REXX Skripte auf eine REST-API umgestellt werden. Dies hat Änderungen an Workflowdefinitionfiles zur Folge. 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 Menüleisteinpunkt „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 am 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 dem 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 für die Rechnungsschreibung individualisiertes 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



Abbildung 5.3.: Beispiel einer Fehlermeldung von zOSMF

das Template und damit alle davon benötigten Dateien auf Syntaxfehler geprüft. Die in 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. Um diese Sprache einzusetzen muss am Zeilenanfang ein „#“ angegeben werden. Diese Sprache bietet einige Features. So ermöglicht diese die dynamische Wertzuweisung von zum Beispiel REXX-Variablen durch Variablen des Templates. Außerdem besteht eine Art von String Verarbeitung. 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 = ${instance-DFH_MQ_QUEUEENAMES})
3 #set ($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_QUEUEENAMES“ Variablen und schreiben in REXX Variablen

In Abbildung 5.5 wird das Ergebnis, welches beim Run erzeugt 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 eine Aussage, um welche Sprache es sich genau handelt gefunden. Somit liegt dem Wissen über diese Sprache nur der Code aus Beispielen der IBM zugrunde.

```

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 passen nicht zu den DATEV e.G. internen Richtlinien. 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, des RACF-Teams. Generell 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. Damit würde das Gewähren der expliziten Rechte zum Starten eines Jobs mit der technischen User Id entfallen und damit die manuelle Arbeit des „Gewährens“. Um jedoch einen „runAsUser“ in der Stepdefinition angeben zu können, muss in der dem Template zugewiesen „Domain“ ein sogenannter „Cloud Security Admin“ hinterlegt sein. Dieser würde sicherstellen, dass nur die für ein Template zugelassenen User dieses Template auch provisionieren dürfen. D.h. es kann beispielweise sichergestellt werden, dass nur DATEV-Rechnungsschreibungs-Entwickler das DATEV-Rechnungsschreibungs Template provisionieren. 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. Dadurch ist es notwendig, die nicht optimale Lösung mit der Jobkarte¹³ einzusetzen. An diesen beiden Fällen ist zu erkennen, dass das Rechtekonzept noch nicht für einen firmenweiten Einsatz bei DATEV e.G. ausgelegt ist und noch überarbeitet und angepasst werden muss. Dies ist jedoch explizit nicht Bestandteil dieser Arbeit.

¹³Siehe Absatz 5.3.1

5.6. Interviews

Um ein allgemeines Meinungsbild zu „IBM Cloud Provisioning and Management for z/OS“ und einen möglichen Nutzwert bei den Stakeholdern, also den Administratorenteams, den Entwicklern und dem Technologiestrategieteam, zu erfassen wurden Interviews mit einzelnen Vertretern dieses Teams durchgeführt. Sowohl der Fragenkatalog, als auch die ausgefüllten und digitalisierten Fragebögen sind im Anhang A.5 zu finden. Diese 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 manuellem 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 bei z/OSPT die Komplexität des zu erstellenden dynamischen Templates ab. Dieser Effekt wird durch fehlende Toolunterstützung und dem dadurch fehlenden Syntaxhighlighting beim Editieren der Templatedateien bzw. der Workflowdefinitionsdateien verstärkt. Nach der Hürde des Einarbeitungsaufwandes und Eingewöhnung in das Editieren von Template Dateien und der Workflowdefinitionsdateien stehe einer aufwandssparenden Provisionierung mittels „IBM Cloud Provisioning and Management for z/OS“ nichts im Wege.

5.6.2. Db2 Administratoren

Das ganze „IBM Cloud Provisioning and Management for z/OS“-Tool wird als sehr mächtig, aber komplex beschrieben. Im Vergleich dazu funktioniere der momentan etablierte Bereitstellungsprozess sehr gut, da dieser bereits lange eingesetzt werde. Jedoch könnten lange Wartezeiten, die durch die vielen Abhängigkeiten von Personen und Abteilungen zustande kommen, durch einen automatisierten Ablauf mittels des Toolkits eliminiert werden. Der durch z/OSMF ermöglichte Prozess zeige zwar, dass eine Automatisierung in diesem Bereich möglich ist, aber auch 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 „IBM Cloud Provisioning and Management for z/OS“ ermöglichte automatisiertes Provisionieren 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 einen 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 bei der Erstellung eines Templates nicht zu vernachlässigen. Beim z/OSPT Lösungsansatz wird kritisiert, dass es sich 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 „IBM Cloud Provisioning and Management for z/OS“-Tool im Vergleich zur momentanen Cloud Foundry Lösung nicht einfach genug zu verwenden sei. Wenn diese Probleme behoben werden können, könnten sich die IBM MQ Administratoren vorstellen, IBM MQ Ressourcen mittels „IBM Cloud Provisioning and Management for z/OS“ zu verwalten. Dabei sei zu beachten, dass erst noch eigene Erfahrungen mit dem Tool 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, da sie Änderungen direkt in der Variableinputfile vornehmen müssen. 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 verzichtet werden kann, und statt dessen mittels Jenkins Build Pipeline oder dem DATEV „Marktplatz“ gearbeitet 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 zu erreichen. Es stehe dem Sammeln von Erfahrungen mit dem Prozess und dem kompletten Toolkit nichts im Wege. Für die Zukunft könne man sich die Nutzung auch für die Qualitätsicherungs- und Produktionsstage, um dort beispielsweise CICS-Instanzen horizontal zu skalieren, vorstellen.

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 von „IBM Cloud Provisioning and Management for z/OS“. 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 Technologiestrategie 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 werde 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. Der dadurch komplexere Aufbau, der entstehende Erstaufwand bei der Erstellung solcher Templates und die damit verbundene steile Lernkurve hat eine abschreckende Wirkung.

Trotz dieser abschreckenden Wirkung sind auch die Administratorenteams bereit, bei Bereitstellung der notwendigen Kapazitäten, 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 eigene 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 use-case im Abschnitt 5.4.2. Das dort genannte Beispiel einer CICS-Instanz und eindeutigen Application IDs wird hier aufgegriffen. Eine Möglichkeit diese 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 und es kommen Images zum Einsatz. Dadurch fällt das Kopieren des Template für den Mitarbeiter weg, dieser kann mittels des Kommandozeileninterfaces ein z/OSPT-Image bauen und daraus einen z/OSPT-Container erzeugen. Das Kommandozeileninterface hat einen weiteren Vorteil. Mit dessen Hilfe können Arbeitsschritte für die Provisionierung der Middleware über groovy 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 würde weiterhin die z/OSMF Oberfläche benötigt. Daran ist zu erkennen, dass von Seiten von z/OSMF beziehungsweise von IBM Verbesserungsmöglichkeiten bestehen.

Diese technische Umsetzung ermöglicht in Zukunft den in Diagramm 6.1 dargestellten Bereitstellungsprozess. Es ist zu erkennen, dass Verantwortung für die Bereitstellung einer Laufzeitumgebung in der Entwicklungsphase von den Administratorenteams an die Entwicklerteams übertragen wird. Dadurch wird Kommunikationsaufwand eingespart und einem Entwickler steht binnen weniger Minuten eine funktionsfähige Laufzeitumgebung in der Entwicklungsphase für seine legacy z/OS Anwendung zur Verfügung. Bei Problemen oder Beratungswunsch unterstützen die Administratorenteams weiterhin. Der Hauptaufwand der Umsetzung dieser Lösung liegt bei den Administratorenteams. Der Effizienzgewinn stellt sich ein, wenn für die wichtigsten Anwendungen Templates existieren, die von den Entwicklern jederzeit genutzt werden können, um individuelle Umgebungen zu nutzen.

Ist all dies umgesetzt, kann in einem nächsten Schritt der cloud native Aspekt des automatisierten Deployments einer Anwendung inklusive Laufzeitumgebung zwischen Stages bis hin zur Produktion auch für z/OS Anwendungen in Betracht gezogen werden.

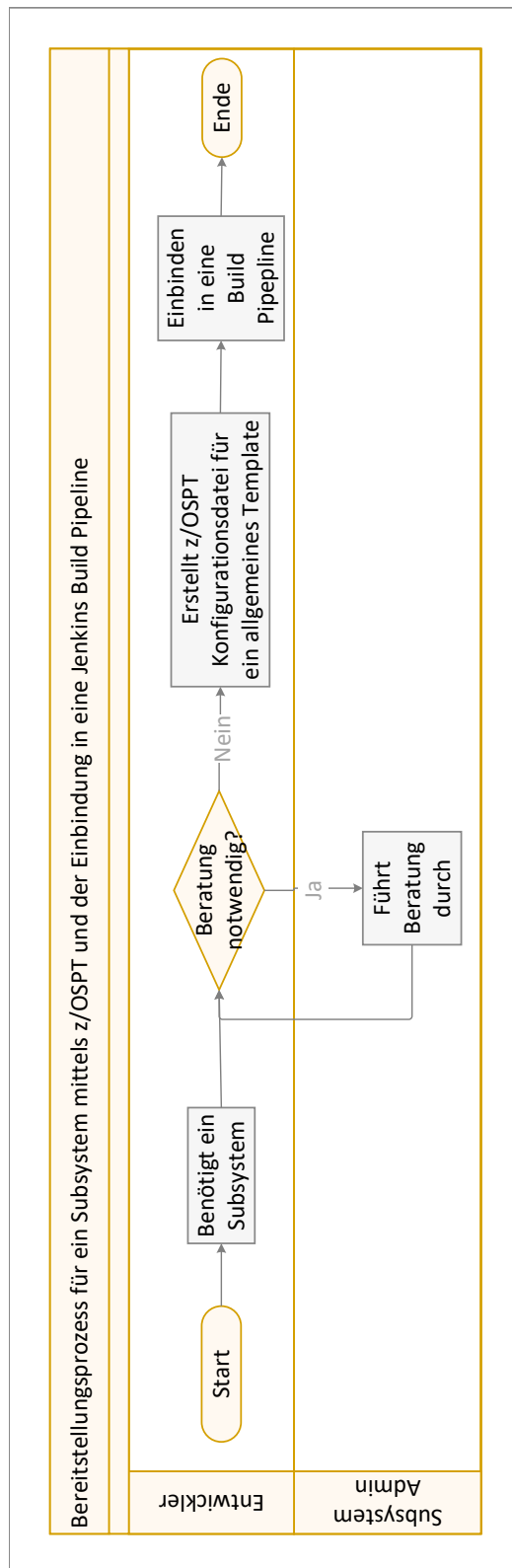


Abbildung 6.1.: Bereitstellungsprozess eines Subsystems mittels einer z/OSPT Konfigurationsdatei

Kapitel 7.

Zusammenfassung und Fazit

Zusammenfassend lässt sich sagen, dass es generell möglich ist mit „IBM Cloud Provisioning and Management for z/OS“ Laufzeitumgebungen für legacy z/OS Anwendungen automatisiert bereitzustellen. Wie aus Tabelle 7.1 zu erkennen ist, ist es in einem gewissen Grad auch möglich damit den Bereitstellungsprozess für z/OS Anwendungen bei DATEV e.G. an den cloud native Prozess anzunähern.

	„IBM Cloud Provisioning and Management for z/OS“	cloud native
Produkt-Teams	nein	ja
automatisierte Bereitstellung von Laufzeitumgebungen in der:		
Entwicklungsstage	ja	ja
Qualitätssicherungsstage	nein	ja
Produktionsstage	nein	ja
CI/CD-Pipeline Unterstützung	ja, mit z/OSPT	ja

Tabelle 7.1.: Vergleich von „IBM Cloud Provisioning and Management for z/OS“ und cloud native im Bezug auf ihren Bereitstellungsprozess

Die Annäherung beschränkt sich jedoch nur auf eine automatisierte Provisionierung von Laufzeitumgebungen in der Entwicklungsstage. Es kann kein klassisches Product-Team-Vorgehen umgesetzt werden, sondern auch mit den Einsatz von „IBM Cloud Provisioning and Management for z/OS“ bleibt die Verwaltung und Überwachung der Middleware bei den Administratorenteams.

Wird der z/OSMF Lösungsansatz genauer betrachtet, dann ist zu erkennen, dass dieser durch den Abbau der Kommunikation zwischen den Abteilungen und nur einmaligem Erstellen der Skripte weniger fehleranfällig und effizienter als der momentan etablierte Prozess ist. 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 e.G. internen „Marktplatz“ für Cloud Lösungen ermöglichen. Um diese Ziele zu erreichen, muss noch viel Aufwand in die Gestaltung des Templates gesteckt und Best Practices erarbeitet werden. Zusätzlich müsste ein sogenannter „Service Broker“ für die Einbindung der einzelnen Subsysteme in den „Marktplatz“ implementiert werden. z/OSPT ist dadurch dem cloud native Prozess näher als z/OSMF.

Beide Lösungsansätze erzeugen bei den Stakeholdern, also den Entwicklerteams und den Administratorenteams einen Mehrwert und werden akzeptiert. Allgemein lässt sich sagen, dass die Nutzung von „IBM Cloud Provisioning and Management for z/OS“ ein kleiner Baustein in Richtung von DevOps und einer CI/CD-Pipeline für den Mainframe sein kann. Dieser Baustein bezieht sich nur auf den Aspekt von automatisierter Bereitstellung von isolierten Testumgebungen. Dieser Aspekt ist jedoch essenziell für den Einsatz einer Build Pipeline. So kann dieser kleine aber wichtige Baustein zu einer besseren Entwicklungseffizienz beitragen. Außerdem hilft dieser Schritt, um dem Image eines veralteten Systems mit veraltetem langsamen Prozesses zu entkommen.

Anhang A.

Anhang

A.1. Verwendete Versionen der z/OS Komponenten im Rahmen dieser Arbeit

- CICS Transaction Server 5.4
- IBM MQ Version 9.1.0.0
- Db2 v11 item z/OS 2.3 (inkl. IBM Cloud Provisioning and Management for z/OS, z/OSMF, z/OSPT)

A.2. 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.3. IT workload distribution worldwide in 2018 and 2020, by cloud type

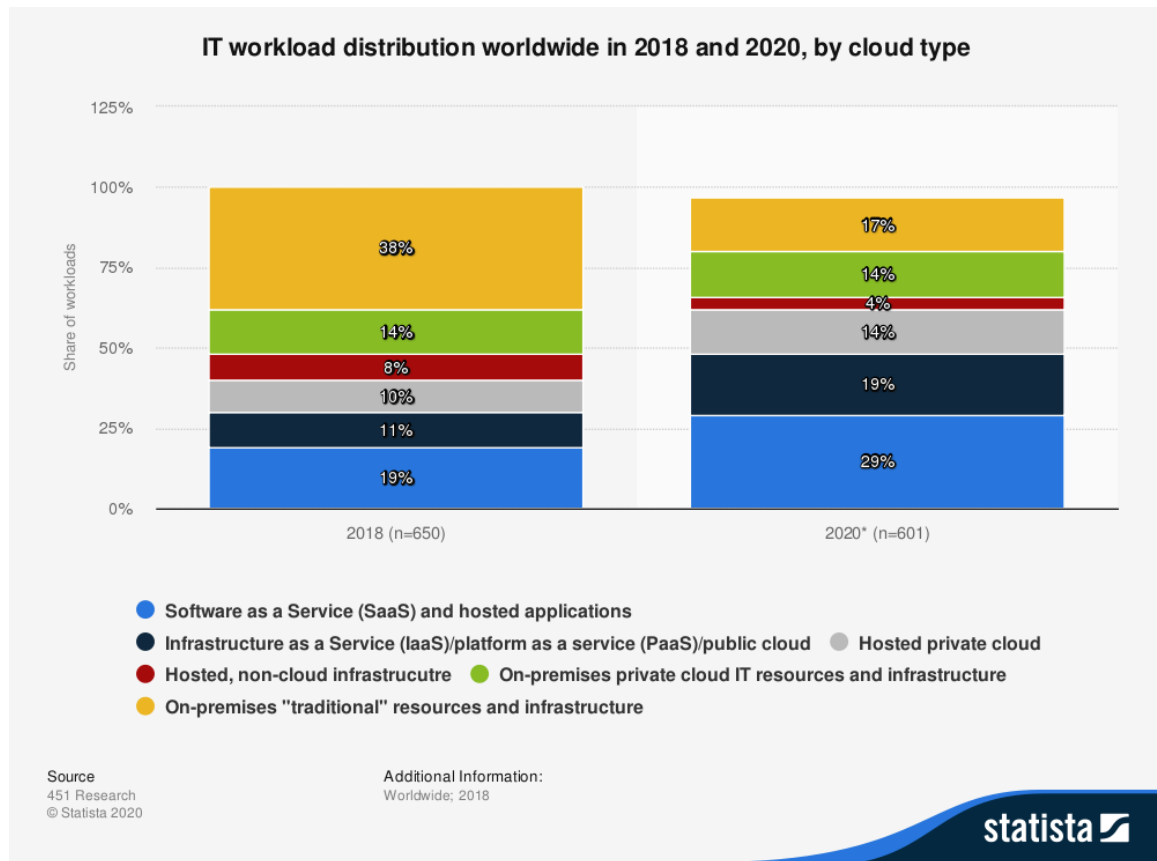


Abbildung A.1.: Weltweiter IT Workload im Jahr 2018 und als Vorhersage im Jahr 2020 bei Cloudtyp

A.4. Fragen an die IBM

Krug, David

Von: Krug, David
Gesendet: Freitag, 28. Februar 2020 10:36
An: 'Marcel Amrein'
Betreff: AW: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Amrein,

vielen Dank für Ihre Antwort.

Mit freundlichem Gruß

David Krug
Accounting Datenmanagement und Archivservices | T1331

DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg

-----Ursprüngliche Nachricht-----

Von: Marcel Amrein <marcel.amrein@de.ibm.com>
Gesendet: Donnerstag, 27. Februar 2020 17:21
An: Krug, David <David.Krug@DATEV.DE>
Cc: Tobias Leicher <tobias.leicher@de.ibm.com>
Betreff: Re: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Krug,
gerne nehme ich zu Ihren Fragen kurz Stellung:

Ich fange mit der zweiten an:

>> 2. Was ist die Motivation der IBM, hier ein Tool bereitzustellen? />>

IBM ist dabei, den Mainframe Cloud-fähig zu machen. Dies bedeutet vor allem und zuerst, dass Mainframe Komponenten sich in eine - hybride, Plattform-neutrale bzw. -übergreifende- Cloud-Architektur "nahtlos" einfügen lassen müssen, was "von Haus aus" nicht der Fall ist. Zu den ersten Ansätzen gehört dabei, Entwicklern das Provisionieren von Anwendungs-Runtime-Umgebungen in Eigenregie ("Self-Service") zu ermöglichen, und hierbei stehen CICS Regions als die wichtigste und am weitesten verbreitete traditionelle Mainframe-Runtime an vorderster Stelle.

die andere Frage:

>> 1. Wie ist aus Ihrer IBM Sicht und Ihre Erfahrungen mit z/OS Kunden die Einschätzung zum IBM z/OS Cloud Provisioning? />>

Ich persönlich habe in diesem Zusammenhang fast nur mir CICS Kunden zu tun,

und halte "CICS" auch für das interessanteste Anwendungsgebiet für Middleware Cloud Provisioning- das schlägt sich übrigens auch darin nieder, dass das CICS Entwicklungslabor die Führung bei der Gestaltung und Weiterentwicklung des "Cloud Provisioning Toolkits" übernommen hat, welches eine wichtige Zwischenschicht in der Gesamtarchitektur darstellt. Traditionell sind die meisten IBM Mainframe Betreiber eher konservativ aufgestellt, und die Systemadministratoren, die über Verprobung und Einsatz dieses Tools befinden, sind diesem zunächst überwiegend skeptisch begegnet, "weil man so etwas bisher nicht vermisst hat und auch mit konventionellen Mitteln in sehr kurzer Zeit z.B. eine CICS Region neu erstellen kann". Mittlerweile ist jedoch zu beobachten, dass in der Mehrzahl der Unternehmen, bei denen IBM Z als strategische Plattform gesetzt ist, starkes Interesse an der Cloud ProvisioningThematik besteht, weil nun eben die "Botschaft" angekommen ist, dass es darum geht, als Mainframe-Plattform im künftigen (Hybrid) Cloud Geschehen mitzuwirken - und nicht etwa darum, die Anforderungen der Vergangenheit mit einem neuen Tool zu bedienen, Dies schlägt sich darin nieder, dass bei Kundenveranstaltungen wie zuletzt der Z Technical University (Mai 2019, Berlin) die Cloud ProvisioningThematik stark nachgefragt war, auch das entsprechende Hands-On Lab, welches ich bereit gestellt hatte, war gut gebucht, und seither haben verschiedene weitere Z Kunden begonnen, sich mit der Thematik zu beschäftigen.

Mit freundlichen Grüßen / Kind regards

Marcel Amrein

Senior Technical Sales Professional (MQ and CICS)
Certified IT Specialist
IBM Global Markets - Cloud Sales
B462

Phone:	+49-7034-643 0038	IBM Deutschland GmbH
Mobile:	+49-160 901 77 458	IBM-Allee 1
Email:	marcel.amrein@de.ibm.com	71139 Ehningen

Germany

IBM Data Privacy
Statement

IBM Deutschland
GmbH /
Vorsitzender des
Aufsichtsrats:
Martin Jetter
Geschäftsführung:
Gregor Pillen
(Vorsitzender),
Agnes Heftberger,
Norbert Janzen,
Markus Koerner,
Christian Noll,
Nicole Reimer
Sitz der
Gesellschaft:
Ehningen /
Registergericht:
Amtsgericht
Stuttgart, HRB
14562 /
WEEE-Reg.-Nr. DE
99369940

From: "Krug, David" <David.Krug@DATEV.DE>
To: "'marcel.amrein@de.ibm.com'" <marcel.amrein@de.ibm.com>
Date: 27/02/2020 15:42
Subject: [EXTERNAL] Bachelorarbeit im Rahmen von IBM z/OS Cloud
Provisioning

Hallo Herr Amrein,

ich schreibe eine Bachelorarbeit mit dem Thema „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.“.

Dies bezüglich habe ich folgende Fragen an Sie:

1. Wie ist aus Ihrer IBM Sicht und Ihre Erfahrungen mit z/OS Kunden die Einschätzung zum IBM z/OS Cloud Provisioning?
2. Was ist die Motivation der IBM, hier ein Tool bereitzustellen?

Ich bedanke mich im Voraus für Ihre Antworten.

Mit freundlichem Gruß
David Krug
Accounting Datenmanagement und Archivservices | T1331
DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg

Signatur

Diese E-Mail wurde mit einem Zertifikat der DATEV eG signiert. Damit können Sie sicher sein, dass die Nachricht so von uns gesendet wurde. Wenn Sie eine Meldung erhalten, dass die Signatur ungültig ist oder nicht geprüft werden kann, fehlt das Zertifikat zu dieser Signatur auf Ihrem Rechner. Informationen zu Zertifikaten und zur digitalen Signatur finden Sie unter <https://www.datev.de/zertifikate> im Internet.

Datenschutz

Informationen zum Umgang mit Ihren personenbezogenen Daten bei DATEV finden Sie unter <https://www.datev.de/dsgvo-information>

DATEV eG
90329 Nürnberg

Telefon +49 911 319-0

E-Mail: info@datev.de
Internet: <https://www.datev.de>

Sitz: 90429 Nürnberg,
Paumgartnerstraße 6-14
Registergericht Nürnberg, GenReg
Nr. 70

Vorstand

Dr. Robert Mayr (Vorsitzender)
Eckhard Schwarzer (stellv.
Vorsitzender)
Julia Bangerth
Prof. Dr. Peter Krug
Diana Windmeißer

Vorsitzender des Aufsichtsrates:
Nicolas Hofmann

Krug, David

Von: Krug, David
Gesendet: Freitag, 28. Februar 2020 10:36
An: 'Tobias Leicher'
Betreff: AW: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Leicher,

vielen Dank für Ihre Antwort.

Mit freundlichem Gruß

David Krug
Accounting Datenmanagement und Archivservices | T1331

DATEV eG
Standort: Fürther Straße 111
Telefon: +49(911)319-42873
E-Mail: david.krug@datev.de
Postanschrift: DATEV eG, 90329 Nürnberg
-----Ursprüngliche Nachricht-----

Von: Tobias Leicher <tobias.leicher@de.ibm.com>
Gesendet: Donnerstag, 27. Februar 2020 17:02
An: Krug, David <David.Krug@DATEV.DE>
Betreff: Re: Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Krug,

Gerade im Umfeld von hochtransaktionalen Enterprise Systemen wie IBM Z sind Kunden weltweit sehr zurückhaltend Cloud Provisionierungsmodelle einzusetzen. Da aber auch hier haben Entwickler in agilen Projekten einen hohen Bedarf an automatisierter Bereitstellung von Umgebungen, nicht nur in einem Developer Marketplace, vor allem in automatisierten Test und Build Pipelines. Viele Kunden haben daher im Moment Interesse, jedoch warten viele hier auf die ersten Erfahrungen von anderen.

IBMs Strategie für Kunden ist eine hybride Multicloud und diese beinhaltet nicht nur x86 Systeme, sondern je nach Anwendungszweck die bestmögliche Hard- und Software. IBM Z ist für hochtransaktionale und hoch regulierte Aufgaben nach wie vor IBMs premium Plattform und daher auch ein essenzieller Teil unserer Cloud Strategie. IBM Z findet hierin sowohl als private Cloud Offering mit zOSMF, zospt und IBM Z Cloud Broker, als auch mit dem hyperprotect Portfolio in der public Cloud einen Platz.

Viele Grüße,
Tobias Leicher

Mit freundlichen Grüßen / Kind regards

Tobias Leicher

zClient Architect
zChampion for Modernization
Senior IT Specialist for CICS and zAPI
IBM Systems Software Technical Sales

Phone:
+49-151-15 16 24 89
IBM Deutschland

E-Mail:
tobias.leicher@de.ibm.com
IBM Allee 1

71139 Ehningen

Germany

Vorsitzender des Aufsichtsrats: Martin Jetter
Geschäftsführung: Gregor Pillen (Vorsitzender), Norbert Janzen, Stefan Lutz, Nicole Reimer, Dr. Klaus Seifert, Wolfgang Wendt
Sitz der Gesellschaft: Ehningen / Registergericht: Amtsgericht Stuttgart, HRB 14562 / WEEE-Reg.-Nr. DE 99369940

From: "Krug, David" <David.Krug@DATEV.DE>
To: "'tobias.leicher@de.ibm.com'" <tobias.leicher@de.ibm.com>
Date: 27/02/2020 15:43
Subject: [EXTERNAL] Bachelorarbeit im Rahmen von IBM z/OS Cloud Provisioning

Hallo Herr Leicher,

ich schreibe eine Bachelorarbeit mit dem Thema ?Automatisierte Provisionierungsmechanismen für Laufzeitumgebungen von Legacy z/OS Anwendungen mit ?IBM Cloud Provisioning and Managment for z/OS? am Beispiel der ?Rechnungsschreibung? bei DATEV e.G.?.
Dies bezüglich habe ich folgende Fragen an Sie:

1. Wie ist aus Ihrer IBM Sicht und Ihre Erfahrungen mit z/OS Kunden die Einschätzung zum IBM z/OS Cloud Provisioning?
2. Was ist die Motivation der IBM, hier ein Tool bereitzustellen?

Ich bedanke mich im Voraus für Ihre Antworten.

Mit freundlichem Gruß

David Krug

Accounting Datenmanagement und Archivservices | T1331

DATEV eG

Standort: Fürther Straße 111

Telefon: +49(911)319-42873

E-Mail: david.krug@datev.de

Postanschrift: DATEV eG, 90329 Nürnberg

Signatur

Diese E-Mail wurde mit einem Zertifikat der DATEV eG signiert. Damit können Sie sicher sein, dass die Nachricht so von uns gesendet wurde. Wenn Sie eine Meldung erhalten, dass die Signatur ungültig ist oder nicht geprüft werden kann, fehlt das Zertifikat zu dieser Signatur auf Ihrem Rechner. Informationen zu Zertifikaten und zur digitalen Signatur finden Sie unter <https://www.datev.de/zertifikate> im Internet.

Datenschutz

Informationen zum Umgang mit Ihren personenbezogenen Daten bei DATEV finden Sie unter <https://www.datev.de/dsgvo-information>

DATEV eG

90329 Nürnberg

Telefon +49 911 319-0

E-Mail: info@datev.de

Internet: <https://www.datev.de>

Sitz: 90429 Nürnberg, Paumgartnerstraße 6-14

Registergericht Nürnberg, GenReg Nr. 70

Vorstand

Dr. Robert Mayr (Vorsitzender)

Eckhard Schwarzer (stellv. Vorsitzender)

Julia Bangerth

Prof. Dr. Peter Krug

Diana Windmeißer

Vorsitzender des Aufsichtsrates: Nicolas Hofmann

A.5. 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.6. 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>

```

A.7. Produktstammdaten Tabellen Data Definition Language

```

1  CREATE TABLE PSSSCHEMA.TPAUSPSSBASELBART
2      (PARTITIONID INTEGER
3
4      WITH DEFAULT 1
5      ,PID INTEGER
6
7      ,AUSPRAEGUNG SMALLINT
8
9      ,GUELTIGAB DATE
10
11     ,LFDNR INTEGER
12
13     WITH DEFAULT
14     ,GUELTIGBIS DATE
15
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          NOT NULL
93
94      ,MAXNUM INTEGER
95
96          NOT NULL
97      ,MAXNUBEZ CHARACTER(42) FOR SBCS DATA
98
99          NOT NULL
100      WITH DEFAULT "X"
101      ,BEZEICHNUNG VARCHAR(100) FOR SBCS DATA
102
103          NOT NULL
104      WITH DEFAULT "X"
105      ,CONSTRAINT MAXNUMID PRIMARY KEY
106      (MAXNUMID
107      )
108      )
109      IN DATABASE PSSBAPRV
110      APPEND NO
111      NOT VOLATILE CARDINALITY
112      DATA CAPTURE NONE
113      AUDIT NONE
114      CCSID EBCDIC;
115
116      COMMENT ON TABLE PSSSCHEMA.TMAXNUM
117          IS "maximale_□Nummer" ;

```

```

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
233          WITH DEFAULT 1
234          ,ARTNR INTEGER
235
236          ,PREISTYPID SMALLINT
237

```

NOT NULL

NOT NULL

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
373  WITH DEFAULT "1966-02-14"
374      ,OPDATBEN CHARACTER(1) FOR SBCS DATA
375
376      NOT NULL
377  WITH DEFAULT "J"
378      ,AUSDIENSTREL CHARACTER(1) FOR SBCS DATA
379
380      NOT NULL
381  WITH DEFAULT "N"
382      ,VERTRIEBSREL CHARACTER(1) FOR SBCS DATA
383
384      NOT NULL
385  WITH DEFAULT "N"
386      ,VERTRELDAT DATE
387
388      WITH DEFAULT NULL
389      ,KOMMASTELLEN INTEGER
390
391      WITH DEFAULT NULL
392      ,ERTRNR INTEGER
393
394      NOT NULL
395      ,GFEDNR INTEGER
396
397      NOT NULL
398  WITH DEFAULT NULL
399      ,UPLONR INTEGER
400
401      NOT NULL
402  WITH DEFAULT NULL
403      ,MWSTEUERSATZ INTEGER
404
405      NOT NULL
406  WITH DEFAULT NULL
407      ,ANZAHL INTEGER
408
409      NOT NULL
410  WITH DEFAULT NULL
411      ,ANZAHL2 INTEGER
412
413      NOT NULL
414  WITH DEFAULT NULL
415      ,ANZAHL3 INTEGER
416
417      NOT NULL
418  WITH DEFAULT NULL
419      ,ANZAHL4 INTEGER
420
421      NOT NULL
422  WITH DEFAULT NULL
423      ,ANZAHL5 INTEGER
424
425      NOT NULL
426  WITH DEFAULT NULL
427      ,ANZAHL6 INTEGER
428
429      NOT NULL
430  WITH DEFAULT NULL
431      ,ANZAHL7 INTEGER
432
433      NOT NULL
434  WITH DEFAULT NULL
435      ,ANZAHL8 INTEGER
436
437      NOT NULL
438  WITH DEFAULT NULL
439      ,ANZAHL9 INTEGER
440
441      NOT NULL
442  WITH DEFAULT NULL
443      ,ANZAHL10 INTEGER
444
445      NOT NULL
446  WITH DEFAULT NULL
447      ,ANZAHL11 INTEGER
448
449      NOT NULL
450  WITH DEFAULT NULL
451      ,ANZAHL12 INTEGER
452
453      NOT NULL
454  WITH DEFAULT NULL
455      ,ANZAHL13 INTEGER
456
457      NOT NULL
458  WITH DEFAULT NULL
459      ,ANZAHL14 INTEGER
460
461      NOT NULL
462  WITH DEFAULT NULL
463      ,ANZAHL15 INTEGER
464
465      NOT NULL
466  WITH DEFAULT NULL
467      ,ANZAHL16 INTEGER
468
469      NOT NULL
470  WITH DEFAULT NULL
471      ,ANZAHL17 INTEGER
472
473      NOT NULL
474  WITH DEFAULT NULL
475      ,ANZAHL18 INTEGER
476
477      NOT NULL
478  WITH DEFAULT NULL
479      ,ANZAHL19 INTEGER
480
481      NOT NULL
482  WITH DEFAULT NULL
483      ,ANZAHL20 INTEGER
484
485      NOT NULL
486  WITH DEFAULT NULL
487      ,ANZAHL21 INTEGER
488
489      NOT NULL
490  WITH DEFAULT NULL
491      ,ANZAHL22 INTEGER
492
493      NOT NULL
494  WITH DEFAULT NULL
495      ,ANZAHL23 INTEGER
496
497      NOT NULL
498  WITH DEFAULT NULL
499      ,ANZAHL24 INTEGER
500
501      NOT NULL
502  WITH DEFAULT NULL
503      ,ANZAHL25 INTEGER
504
505      NOT NULL
506  WITH DEFAULT NULL
507      ,ANZAHL26 INTEGER
508
509      NOT NULL
510  WITH DEFAULT NULL
511      ,ANZAHL27 INTEGER
512
513      NOT NULL
514  WITH DEFAULT NULL
515      ,ANZAHL28 INTEGER
516
517      NOT NULL
518  WITH DEFAULT NULL
519      ,ANZAHL29 INTEGER
520
521      NOT NULL
522  WITH DEFAULT NULL
523      ,ANZAHL30 INTEGER
524
525      NOT NULL
526  WITH DEFAULT NULL
527      ,ANZAHL31 INTEGER
528
529      NOT NULL
530  WITH DEFAULT NULL
531      ,ANZAHL32 INTEGER
532
533      NOT NULL
534  WITH DEFAULT NULL
535      ,ANZAHL33 INTEGER
536
537      NOT NULL
538  WITH DEFAULT NULL
539      ,ANZAHL34 INTEGER
540
541      NOT NULL
542  WITH DEFAULT NULL
543      ,ANZAHL35 INTEGER
544
545      NOT NULL
546  WITH DEFAULT NULL
547      ,ANZAHL36 INTEGER
548
549      NOT NULL
550  WITH DEFAULT NULL
551      ,ANZAHL37 INTEGER
552
553      NOT NULL
554  WITH DEFAULT NULL
555      ,ANZAHL38 INTEGER
556
557      NOT NULL
558  WITH DEFAULT NULL
559      ,ANZAHL39 INTEGER
560
561      NOT NULL
562  WITH DEFAULT NULL
563      ,ANZAHL40 INTEGER
564
565      NOT NULL
566  WITH DEFAULT NULL
567      ,ANZAHL41 INTEGER
568
569      NOT NULL
570  WITH DEFAULT NULL
571      ,ANZAHL42 INTEGER
572
573      NOT NULL
574  WITH DEFAULT NULL
575      ,ANZAHL43 INTEGER
576
577      NOT NULL
578  WITH DEFAULT NULL
579      ,ANZAHL44 INTEGER
580
581      NOT NULL
582  WITH DEFAULT NULL
583      ,ANZAHL45 INTEGER
584
585      NOT NULL
586  WITH DEFAULT NULL
587      ,ANZAHL46 INTEGER
588
589      NOT NULL
590  WITH DEFAULT NULL
591      ,ANZAHL47 INTEGER
592
593      NOT NULL
594  WITH DEFAULT NULL
595      ,ANZAHL48 INTEGER
596
597      NOT NULL
598  WITH DEFAULT NULL
599      ,ANZAHL49 INTEGER
600
601      NOT NULL
602  WITH DEFAULT NULL
603      ,ANZAHL50 INTEGER
604
605      NOT NULL
606  WITH DEFAULT NULL
607      ,ANZAHL51 INTEGER
608
609      NOT NULL
610  WITH DEFAULT NULL
611      ,ANZAHL52 INTEGER
612
613      NOT NULL
614  WITH DEFAULT NULL
615      ,ANZAHL53 INTEGER
616
617      NOT NULL
618  WITH DEFAULT NULL
619      ,ANZAHL54 INTEGER
620
621      NOT NULL
622  WITH DEFAULT NULL
623      ,ANZAHL55 INTEGER
624
625      NOT NULL
626  WITH DEFAULT NULL
627      ,ANZAHL56 INTEGER
628
629      NOT NULL
630  WITH DEFAULT NULL
631      ,ANZAHL57 INTEGER
632
633      NOT NULL
634  WITH DEFAULT NULL
635      ,ANZAHL58 INTEGER
636
637      NOT NULL
638  WITH DEFAULT NULL
639      ,ANZAHL59 INTEGER
640
641      NOT NULL
642  WITH DEFAULT NULL
643      ,ANZAHL60 INTEGER
644
645      NOT NULL
646  WITH DEFAULT NULL
647      ,ANZAHL61 INTEGER
648
649      NOT NULL
650  WITH DEFAULT NULL
651      ,ANZAHL62 INTEGER
652
653      NOT NULL
654  WITH DEFAULT NULL
655      ,ANZAHL63 INTEGER
656
657      NOT NULL
658  WITH DEFAULT NULL
659      ,ANZAHL64 INTEGER
660
661      NOT NULL
662  WITH DEFAULT NULL
663      ,ANZAHL65 INTEGER
664
665      NOT NULL
666  WITH DEFAULT NULL
667      ,ANZAHL66 INTEGER
668
669      NOT NULL
670  WITH DEFAULT NULL
671      ,ANZAHL67 INTEGER
672
673      NOT NULL
674  WITH DEFAULT NULL
675      ,ANZAHL68 INTEGER
676
677      NOT NULL
678  WITH DEFAULT NULL
679      ,ANZAHL69 INTEGER
680
681      NOT NULL
682  WITH DEFAULT NULL
683      ,ANZAHL70 INTEGER
684
685      NOT NULL
686  WITH DEFAULT NULL
687      ,ANZAHL71 INTEGER
688
689      NOT NULL
690  WITH DEFAULT NULL
691      ,ANZAHL72 INTEGER
692
693      NOT NULL
694  WITH DEFAULT NULL
695      ,ANZAHL73 INTEGER
696
697      NOT NULL
698  WITH DEFAULT NULL
699      ,ANZAHL74 INTEGER
700
701      NOT NULL
702  WITH DEFAULT NULL
703      ,ANZAHL75 INTEGER
704
705      NOT NULL
706  WITH DEFAULT NULL
707      ,ANZAHL76 INTEGER
708
709      NOT NULL
710  WITH DEFAULT NULL
711      ,ANZAHL77 INTEGER
712
713      NOT NULL
714  WITH DEFAULT NULL
715      ,ANZAHL78 INTEGER
716
717      NOT NULL
718  WITH DEFAULT NULL
719      ,ANZAHL79 INTEGER
720
721      NOT NULL
722  WITH DEFAULT NULL
723      ,ANZAHL80 INTEGER
724
725      NOT NULL
726  WITH DEFAULT NULL
727      ,ANZAHL81 INTEGER
728
729      NOT NULL
730  WITH DEFAULT NULL
731      ,ANZAHL82 INTEGER
732
733      NOT NULL
734  WITH DEFAULT NULL
735      ,ANZAHL83 INTEGER
736
737      NOT NULL
738  WITH DEFAULT NULL
739      ,ANZAHL84 INTEGER
740
741      NOT NULL
742  WITH DEFAULT NULL
743      ,ANZAHL85 INTEGER
744
745      NOT NULL
746  WITH DEFAULT NULL
747      ,ANZAHL86 INTEGER
748
749      NOT NULL
750  WITH DEFAULT NULL
751      ,ANZAHL87 INTEGER
752
753      NOT NULL
754  WITH DEFAULT NULL
755      ,ANZAHL88 INTEGER
756
757      NOT NULL
758  WITH DEFAULT NULL
759      ,ANZAHL89 INTEGER
760
761      NOT NULL
762  WITH DEFAULT NULL
763      ,ANZAHL90 INTEGER
764
765      NOT NULL
766  WITH DEFAULT NULL
767      ,ANZAHL91 INTEGER
768
769      NOT NULL
770  WITH DEFAULT NULL
771      ,ANZAHL92 INTEGER
772
773      NOT NULL
774  WITH DEFAULT NULL
775      ,ANZAHL93 INTEGER
776
777      NOT NULL
778  WITH DEFAULT NULL
779      ,ANZAHL94 INTEGER
780
781      NOT NULL
782  WITH DEFAULT NULL
783      ,ANZAHL95 INTEGER
784
785      NOT NULL
786  WITH DEFAULT NULL
787      ,ANZAHL96 INTEGER
788
789      NOT NULL
790  WITH DEFAULT NULL
791      ,ANZAHL97 INTEGER
792
793      NOT NULL
794  WITH DEFAULT NULL
795      ,ANZAHL98 INTEGER
796
797      NOT NULL
798  WITH DEFAULT NULL
799      ,ANZAHL99 INTEGER
800
801      NOT NULL
802  WITH DEFAULT NULL
803      ,ANZAHL100 INTEGER
804
805      NOT NULL
806  WITH DEFAULT NULL
807      ,ANZAHL101 INTEGER
808
809      NOT NULL
810  WITH DEFAULT NULL
811      ,ANZAHL102 INTEGER
812
813      NOT NULL
814  WITH DEFAULT NULL
815      ,ANZAHL103 INTEGER
816
817      NOT NULL
818  WITH DEFAULT NULL
819      ,ANZAHL104 INTEGER
820
821      NOT NULL
822  WITH DEFAULT NULL
823      ,ANZAHL105 INTEGER
824
825      NOT NULL
826  WITH DEFAULT NULL
827      ,ANZAHL106 INTEGER
828
829      NOT NULL
830  WITH DEFAULT NULL
831      ,ANZAHL107 INTEGER
832
833      NOT NULL
834  WITH DEFAULT NULL
835      ,ANZAHL108 INTEGER
836
837      NOT NULL
838  WITH DEFAULT NULL
839      ,ANZAHL109 INTEGER
840
841      NOT NULL
842  WITH DEFAULT NULL
843      ,ANZAHL110 INTEGER
844
845      NOT NULL
846  WITH DEFAULT NULL
847      ,ANZAHL111 INTEGER
848
849      NOT NULL
850  WITH DEFAULT NULL
851      ,ANZAHL112 INTEGER
852
853      NOT NULL
854  WITH DEFAULT NULL
855      ,ANZAHL113 INTEGER
856
857      NOT NULL
858  WITH DEFAULT NULL
859      ,ANZAHL114 INTEGER
860
861      NOT NULL
862  WITH DEFAULT NULL
863      ,ANZAHL115 INTEGER
864
865      NOT NULL
866  WITH DEFAULT NULL
867      ,ANZAHL116 INTEGER
868
869      NOT NULL
870  WITH DEFAULT NULL
871      ,ANZAHL117 INTEGER
872
873      NOT NULL
874  WITH DEFAULT NULL
875      ,ANZAHL118 INTEGER
876
877      NOT NULL
878  WITH DEFAULT NULL
879      ,ANZAHL119 INTEGER
880
881      NOT NULL
882  WITH DEFAULT NULL
883      ,ANZAHL120 INTEGER
884
885      NOT NULL
886  WITH DEFAULT NULL
887      ,ANZAHL121 INTEGER
888
889      NOT NULL
890  WITH DEFAULT NULL
891      ,ANZAHL122 INTEGER
892
893      NOT NULL
894  WITH DEFAULT NULL
895      ,ANZAHL123 INTEGER
896
897      NOT NULL
898  WITH DEFAULT NULL
899      ,ANZAHL124 INTEGER
900
901      NOT NULL
902  WITH DEFAULT NULL
903      ,ANZAHL125 INTEGER
904
905      NOT NULL
906  WITH DEFAULT NULL
907      ,ANZAHL126 INTEGER
908
909      NOT NULL
910  WITH DEFAULT NULL
911      ,ANZAHL127 INTEGER
912
913      NOT NULL
914  WITH DEFAULT NULL
915      ,ANZAHL128 INTEGER
916
917      NOT NULL
918  WITH DEFAULT NULL
919      ,ANZAHL129 INTEGER
920
921      NOT NULL
922  WITH DEFAULT NULL
923      ,ANZAHL130 INTEGER
924
925      NOT NULL
926  WITH DEFAULT NULL
927      ,ANZAHL131 INTEGER
928
929      NOT NULL
930  WITH DEFAULT NULL
931      ,ANZAHL132 INTEGER
932
933      NOT NULL
934  WITH DEFAULT NULL
935      ,ANZAHL133 INTEGER
936
937      NOT NULL
938  WITH DEFAULT NULL
939      ,ANZAHL134 INTEGER
940
941      NOT NULL
942  WITH DEFAULT NULL
943      ,ANZAHL135 INTEGER
944
945      NOT NULL
946  WITH DEFAULT NULL
947      ,ANZAHL136 INTEGER
948
949      NOT NULL
950  WITH DEFAULT NULL
951      ,ANZAHL137 INTEGER
952
953      NOT NULL
954  WITH DEFAULT NULL
955      ,ANZAHL138 INTEGER
956
957      NOT NULL
958  WITH DEFAULT NULL
959      ,ANZAHL139 INTEGER
960
961      NOT NULL
962  WITH DEFAULT NULL
963      ,ANZAHL140 INTEGER
964
965      NOT NULL
966  WITH DEFAULT NULL
967      ,ANZAHL141 INTEGER
968
969      NOT NULL
970  WITH DEFAULT NULL
971      ,ANZAHL142 INTEGER
972
973      NOT NULL
974  WITH DEFAULT NULL
975      ,ANZAHL143 INTEGER
976
977      NOT NULL
978  WITH DEFAULT NULL
979      ,ANZAHL144 INTEGER
980
981      NOT NULL
982  WITH DEFAULT NULL
983      ,ANZAHL145 INTEGER
984
985      NOT NULL
986  WITH DEFAULT NULL
987      ,ANZAHL146 INTEGER
988
989      NOT NULL
990  WITH DEFAULT NULL
991      ,ANZAHL147 INTEGER
992
993      NOT NULL
994  WITH DEFAULT NULL
995      ,ANZAHL148 INTEGER
996
997      NOT NULL
998  WITH DEFAULT NULL
999      ,ANZAHL149 INTEGER
1000
1001      NOT NULL
1002  WITH DEFAULT NULL
1003      ,ANZAHL150 INTEGER
1004
1005      NOT NULL
1006  WITH DEFAULT NULL
1007      ,ANZAHL151 INTEGER
1008
1009      NOT NULL
1010  WITH DEFAULT NULL
1011      ,ANZAHL152 INTEGER
1012
1013      NOT NULL
1014  WITH DEFAULT NULL
1015      ,ANZAHL153 INTEGER
1016
1017      NOT NULL
1018  WITH DEFAULT NULL
1019      ,ANZAHL154 INTEGER
1020
1021      NOT NULL
1022  WITH DEFAULT NULL
1023      ,ANZAHL155 INTEGER
1024
1025      NOT NULL
1026  WITH DEFAULT NULL
1027      ,ANZAHL156 INTEGER
1028
1029      NOT NULL
1030  WITH DEFAULT NULL
1031      ,ANZAHL157 INTEGER
1032
1033      NOT NULL
1034  WITH DEFAULT NULL
1035      ,ANZAHL158 INTEGER
1036
1037      NOT NULL
1038  WITH DEFAULT NULL
1039      ,ANZAHL159 INTEGER
1040
1041      NOT NULL
1042  WITH DEFAULT NULL
1043      ,ANZAHL160 INTEGER
1044
1045      NOT NULL
1046  WITH DEFAULT NULL
1047      ,ANZAHL161 INTEGER
1048
1049      NOT NULL
1050  WITH DEFAULT NULL
1051      ,ANZAHL162 INTEGER
1052
1053      NOT NULL
1054  WITH DEFAULT NULL
1055      ,ANZAHL163 INTEGER
1056
1057      NOT NULL
1058  WITH DEFAULT NULL
1059      ,ANZAHL164 INTEGER
1060
1061      NOT NULL
1062  WITH DEFAULT NULL
1063      ,ANZAHL165 INTEGER
1064
1065      NOT NULL
1066  WITH DEFAULT NULL
1067      ,ANZAHL166 INTEGER
1068
1069      NOT NULL
1070  WITH DEFAULT NULL
1071      ,ANZAHL167 INTEGER
1072
1073      NOT NULL
1074  WITH DEFAULT NULL
1075      ,ANZAHL168 INTEGER
1076
1077      NOT NULL
1078  WITH DEFAULT NULL
1079      ,ANZAHL169 INTEGER
1080
1081      NOT NULL
1082  WITH DEFAULT NULL
1083      ,ANZAHL170 INTEGER
1084
1085      NOT NULL
1086  WITH DEFAULT NULL
1087      ,ANZAHL171 INTEGER
1088
1089      NOT NULL
1090  WITH DEFAULT NULL
1091      ,ANZAHL172 INTEGER
1092
1093      NOT NULL
1094  WITH DEFAULT NULL
1095      ,ANZAHL173 INTEGER
1096
1097      NOT NULL
1098  WITH DEFAULT NULL
1099      ,ANZAHL174 INTEGER
1100
1101      NOT NULL
1102  WITH DEFAULT NULL
1103      ,ANZAHL175 INTEGER
1104
1105      NOT NULL
1106  WITH DEFAULT NULL
1107      ,ANZAHL176 INTEGER
1108
1109      NOT NULL
1110  WITH DEFAULT NULL
1111      ,ANZAHL177 INTEGER
1112
1113      NOT NULL
1114  WITH DEFAULT NULL
1115      ,ANZAHL178 INTEGER
1116
1117      NOT NULL
1118  WITH DEFAULT NULL
1119      ,ANZAHL179 INTEGER
1120
1121      NOT NULL
1122  WITH DEFAULT NULL
1123      ,ANZAHL180 INTEGER
1124
1125      NOT NULL
1126  WITH DEFAULT NULL
1127      ,ANZAHL181 INTEGER
1128
1129      NOT NULL
1130  WITH DEFAULT NULL
1131      ,ANZAHL182 INTEGER
1132
1133      NOT NULL
1134  WITH DEFAULT NULL
1135      ,ANZAHL183 INTEGER
1136
1137      NOT NULL
1138  WITH DEFAULT NULL
1139      ,ANZAHL184 INTEGER
1140
1141      NOT NULL
1142  WITH DEFAULT NULL
1143      ,ANZAHL185 INTEGER
1144
1145      NOT NULL
1146  WITH DEFAULT NULL
1147      ,ANZAHL186 INTEGER
1148
1149      NOT NULL
1150  WITH DEFAULT NULL
1151      ,ANZAHL187 INTEGER
1152
1153      NOT NULL
1154  WITH DEFAULT NULL
1155      ,ANZAHL188 INTEGER
1156
1157      NOT NULL
1158  WITH DEFAULT NULL
1159      ,ANZAHL189 INTEGER
1160
1161      NOT NULL
1162  WITH DEFAULT NULL
1163      ,ANZAHL190 INTEGER
1164
1165      NOT NULL
1166  WITH DEFAULT NULL
1167      ,ANZAHL191 INTEGER
1168
1169      NOT NULL
1170  WITH DEFAULT NULL
1171      ,ANZAHL192 INTEGER
1172
1173      NOT NULL
1174  WITH DEFAULT NULL
1175      ,ANZAHL193 INTEGER
1176
1177      NOT NULL
1178  WITH DEFAULT NULL
1179      ,ANZAHL194 INTEGER
1180
1181      NOT NULL
1182  WITH DEFAULT NULL
1183      ,ANZAHL195 INTEGER
1184
1185      NOT NULL
1186  WITH DEFAULT NULL
1187      ,ANZAHL196 INTEGER
1188
1189      NOT NULL
1190  WITH DEFAULT NULL
1191      ,ANZAHL197 INTEGER
1192
1193      NOT NULL
1194  WITH DEFAULT NULL
1195      ,ANZAHL198 INTEGER
1196
1197      NOT NULL
1198  WITH DEFAULT NULL
1199      ,ANZAHL199 INTEGER
1200
1201      NOT NULL
1202  WITH DEFAULT NULL
1203      ,ANZAHL200 INTEGER
1204
1205      NOT NULL
1206  WITH DEFAULT NULL
1207      ,ANZAHL201 INTEGER
1208
1209      NOT NULL
1210  WITH DEFAULT NULL
1211      ,ANZAHL202 INTEGER
1212
1213      NOT NULL
1214  WITH DEFAULT NULL
1215      ,ANZAHL203 INTEGER
1216
1217      NOT NULL
1218  WITH DEFAULT NULL
1219      ,ANZAHL204 INTEGER
1220
1221      NOT NULL
1222  WITH DEFAULT NULL
1223      ,ANZAHL205 INTEGER
1224
1225      NOT NULL
1226  WITH DEFAULT NULL
1227      ,ANZAHL206
```


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;
```


Abbildungsverzeichnis

1.1. Anteil der verwendeten Programmiersprachen auf dem Mainframe bei DATEV eG in Prozent	5
1.2. Cloud Foundry Marketplace bei DATEV e.G.	6
1.3. Auszug aus einem REXX Skript in der ISPF Oberfläche	7
2.1. Abgrenzung von Continuous Integration, Continuous Delivery und Continuous Deployment (Quelle: [http 20b])	11
2.2. Continuous Integration Prozessaufbau (Quelle: [http 20c])	12
2.3. Architekturübersicht über die Subsysteme einer Stage bei DATEV eG	14
2.4. Etablierte Konfigurationsoberfläche am Beispiel bei Änderung einer Transaktion	18
2.5. z/OSMF Willkommens Ansicht	21
2.6. z/OSPT mögliche Kommandozeilenbefehle	23
4.1. Bereitstellungsprozess einer CICS Instanz	32
4.2. Bereitstellungsprozess einer Db2 Datenbank	34
4.3. Bereitstellungsprozess einer IBM MQ Queue	36
5.1. Login Bildschirm der provisionierten DATEV spezifischen CICS-Instanz	49
5.2. Define IBM Queue, am Beispiel einer Trigger Queue	51
5.3. Beispiel einer Fehlermeldung von zOSMF	58
6.1. Bereitstellungsprozess eines Subsystems mittels einer z/OSPT Konfigurationsdatei	67
A.1. Weltweiter It Workload im Jahr 2018 und als Vorhersage im Jahr 2020 bei Cloud-typ	74
1. Job Beispiel, Display einer IBM MQ	118

Tabellenverzeichnis

5.1. Vergleich zwischen z/OSPT und z/OSMF	42
5.2. Zu verändernde Variablen im „cics_getting_started“-Template	43
5.3. Zu verändernde Variablen im „cics_54“-Template	44
5.4. Vergleich der beiden Templates im Bezug auf deren Umfang	45
7.1. Vergleich von „IBM Cloud Provisioning and Management for z/OS“ und cloud native im Bezug auf ihren Bereitstellungsprozess	69

Quellcodeverzeichnis

5.1. Hinzufügen weiterer CSD Gruppen zur Liste der provisionierten CICS-Instanz mittels eines Jobs	47
5.2. Setzen der SIT Parameter durch Auslesen der „DFH_REGION_SITPARAMS“ Variablen	48
5.3. Erstellung einer neuen CSD Gruppe	54
5.4. Auslesen der „DFH_MQ_QUEUEENAMES“ Variablen und schreiben in REXX Variablen	59
5.5. Zur Laufzeit erzeugtes Skript, der Grundlage aus Codeabschnitt 5.4	59
listings/db2provision.xml	92
listings/ddl.txt	93

Literaturverzeichnis

- [Adam 16] R. Adams. *SQL: Der Grundkurs für Ausbildung und Praxis : mit Beispielen in MySQL/MariaDB*. 2016.
- [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.
- [Cass 07] P. Cassier. *System programmer’s guide to Workload manager*. 2007.
- [Ceru 03] P. E. Ceruzzi. *A history of modern computing*. 2003.
- [Fosd 05] H. Fosdick. *Rexx programmer’s reference*. 2005.
- [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.bsi.bund.de/DE/Themen/DigitaleGesellschaft/CloudComputing/Grundlagen/CloudComputingGrundlagen.html>”. 23.2.2020.
- [http 20g] “<https://www.cloudcomputing-insider.de/was-ist-cloud-foundry-a-615766/>”. 23.2.2020.

- [http 20h] “<https://www.cloudcomputing-insider.de/was-ist-cloud-native-a-669681/>”. 23.2.2020.
- [http 20i] “<https://www.datev.de/web/de/m/ueber-datev/das-unternehmen/kurzprofil/>”. 27.2.2020.
- [http 20j] “<https://www.egovernment-computing.de/was-ist-ein-legacy-system-a-802283/>”. 22.2.2020.
- [http 20k] “<https://www.fintechfutures.com/2018/11/ibm-shows-off-three-new-bank-clients/>”. 25.2.2020.
- [http 20l] “<https://www.hochschulkompass.de/home.html>”. 09.02.2020.
- [http 20m] “<https://www.ibm.com/de-de/cloud/learn/iaas-paas-saas>”. 25.2.2020.
- [http 20n] “https://www.ibm.com/support/knowledgecenter/SSLTBW_2.4.0/com.ibm.zos.v2r4.izua700/izupr”. 26.2.2020.
- [http 20o] “https://www.ibm.com/support/knowledgecenter/SSXH44_1.1.0/zospt/cics/zospt-cics-properties.html”. 26.2.2020.
- [Kara 08] S. Karan, F. Rui, L. Oerjan, M. Bob, P. Rita, and R. Paul. *ABCs of z/OS system programming, Volume 6*. 2008.
- [Keit 16] W. Keith, P. Gary, and S. Hiren. *IBM Cloud Provisioning and Management for z/OS: An Introduction*. 2016.
- [Kim 14] G. Kim. *The Phoenix project: A novel about IT, DevOps, and helping your business win*. 2014.
- [Last 17] B. Laster. *Continuous Integration vs. Continuous Delivery vs. Continuous Deployment*. 2017.
- [Rott 18] R. J. T. Rotthove. *IBM z/OS Management Facility V2R3*. 2018.
- [Stee 03] B. Steegmans. *DB2 for z/OS and OS/390: Ready for Java*. 2003.
- [Vohr 16] D. Vohra. *Pro Docker*. 2016.

Glossar

B

Batch Batch beziehungsweise Batch-Verarbeitung bezeichnet in der IT die sog. „Stapelverarbeitung“. Das heißt, dass Programme mit minimalem menschlichen Eingreifen nacheinander abgearbeitet werden. Beispielsweise meist zu einer vorher festgelegten Zeit, gesteuert wird dies über sog. „Scheduling“-Systeme. Zum Beispiel wird einmal am Tag zu einer ganz bestimmten Uhrzeit die tägliche Bewertung¹ der DATEV Rechnungsschreibung durchgeführt. Die auszuführenden Programme laufen in sogenannten „Batch-Jobs“. [?]. 13

Batch Job In einem BatchJob, in dieser Arbeit wird „Job“ gleichbedeutend verwendet, wird dem System mitgeteilt welches Programm mit welchen Ein- und Ausgabedateien und Parametern gestartet werden soll. Die Skriptsprache, die diese Jobs definiert, ist im IBM-Mainframe-Umfeld die sog. „Job Control Language“, kurz JCL. Die drei Grundbausteine der JCL werden im Folgenden beschrieben.

Zunächst ist „JOB“ zu nennen, auch Jobkarte genannt. Hier werden der Name des Jobs, Berechnungsinformationen, maximal zur Verfügung stehende CPU-Zeit und weitere Job-weite Parameter gesetzt. Im Beispiel 1 Zeilennummer eins bis drei.

Innerhalb eines Jobs wird mit Hilfe des „EXEC“ Befehls dem System mitgeteilt, welches Programm gestartet werden soll. Es können mehrere „EXEC“ Befehle in einem Job vorkommen, dabei wird jeder einzelne als sogenannter „Job step“ bezeichnet. Dabei können dem Programm neben den Ein-/Ausgabedateien auch weitere Parameter übergeben werden. Im Beispiel 1 Zeilennummer zehn.

Als letztes ist der „DD“ Baustein zu nennen. „DD“ steht für Data Definition. Ein DD-Statement verknüpft den sog. DD-Namen mit einer Datei oder einem I/O Gerät und ist somit ein Alias für diese. Ein „DD“ Baustein ist immer an ein „EXEC“ Befehl gebunden. Einem „EXEC“ können mehrere „DD“ Bausteine zugeordnet sein. Im Beispiel 1 Zeilennummer 12 bis 15 und 19. Hier ist auch zu sehen, dass Daten auch Inline an ein DD-Statement übergeben werden kann. [?] . 4

C

¹Beschreibung in Absatz 4.3.1

VSAM Virtual Storage Access Method, spezielle z/OS Dateart, die schnelle I/O-Zugriffe ermöglicht. (Quelle: [?, S. 2-4]. **13**