



# Jenkins ePortfolio

in

Software-Engineering

von

Michaela Fleig

am

20. Mai 2020

## Agenda

<b>1. Continuous Integration .....</b>	<b>3</b>
<b>1.1. Continuous Integration als Teil von CI/CD .....</b>	<b>3</b>
<b>1.2. CI-Lifecycle.....</b>	<b>3</b>
<b>1.3. Deployment-Pipeline .....</b>	<b>4</b>
<b>2. Hudson/Jenkins .....</b>	<b>5</b>
<b>2.1. Jenkins als Tool für CI .....</b>	<b>5</b>
<b>2.2. Installation .....</b>	<b>5</b>
<b>2.3. Einen Job für lokale Dateien erstellen .....</b>	<b>7</b>
<b>2.4. Einen Job für remote Quellen auf GitHub erstellen .....</b>	<b>9</b>
<b>2.5. Plugins und Tipps .....</b>	<b>11</b>
<b>2.6. Eine Pipeline mit Blue Ocean erstellen .....</b>	<b>11</b>

## 1. Continuous Integration

### 1.1. Continuous Integration als Teil von CI/CD

Continuous Integration/Continuous Development/Continuous Deployment (CI/CD) beschreibt den Lauf der Anwendung von seiner Erstellung beim Entwickler bis zur Bereitstellung beim Kunden. Meistens schreiben mehrere Entwickler an einer Anwendung, oftmals zeitgleich, besonders wenn der Release-Termin näher rückt. Der Einsatz von CI/CD besteht daraus, den erstellten Quellcode auf einen Archiv-Speicher, auch Repository genannt, zu speichern. Anschließend kann auf diesem Quellcode weiterentwickelt werden, der danach dem Kunden bereit gestellt werden kann.

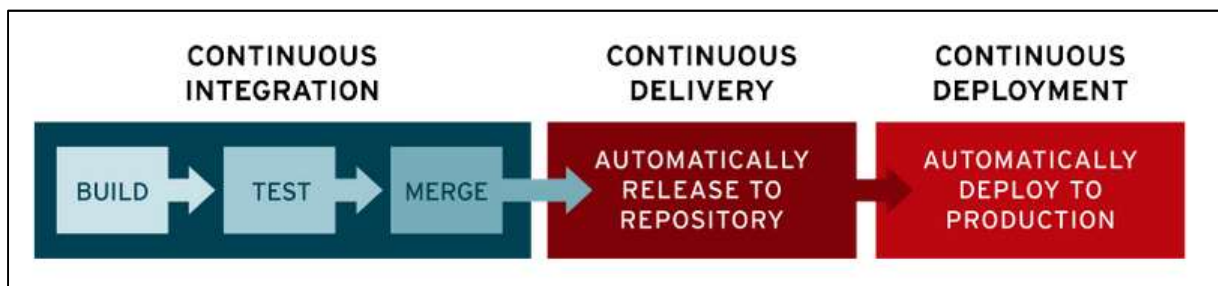


Abbildung 1: Die Phasen der Continuous Integration, der - Delivery und des – Deployments

Abbildung 1 zeigt die Continuous Integration, die aus einem Build der Code-Teile besteht, die der Entwickler lokal erstellt hat. Diese werden auf grobe Syntaxfehler getestet, anschließend gegen Tests geprüft, die ein Entwickler zuvor geschrieben hatte. Wenn der Test erfolgreich besteht, wird dieser Code auf einem Staging System gespeichert. Wenn der Test fehlschlägt, wird der Benutzer über eine Mail benachrichtigt.

Continuous Delivery beschreibt die Übernahme von Code vom Staging System zum Archiv-Speicher. Damit werden weiteren Entwicklern der funktionierende Code zur weiteren Verwendung zur Verfügung gestellt.

Continuous Deployment beschreibt die Freigabe der Dateien an den Kunden, an das Produktivsystem.

Der gesamte Prozess soll die Belastung und den zeitlichen Aufwand der Entwickler minimieren, die für manuelle Tests nötig sind. Eine schnellere Entwicklung und Reaktion auf Kollisionen werden ebenfalls erhofft. Nachteilig wirken sich der hohe Anfangsaufwand und die –kosten durch das Schreiben von automatisierten Tests zur Einführung der CI/CD aus.

### 1.2. CI-Lifecycle

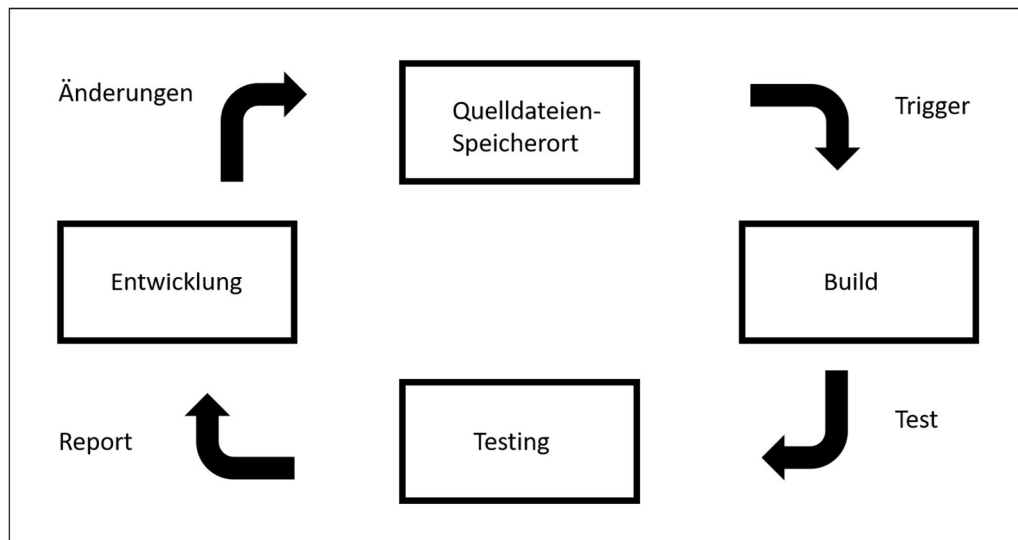


Abbildung 2: Der CI-Lifecycle

Von der Entwicklung und Änderung der Quelldateien wird das CI zu einem Build getriggert. Danach werden die Tests durchgeführt, insofern welche vorhanden sind. Der abschließende Report wird direkt an den Entwickler gesandt. Dieser wird darin informiert, was das CI-System ausgeführt hat, ob der Build und die Tests erfolgreich beendet wurden oder welche Fehler vorhanden sind.

### 1.3. Deployment-Pipeline

Der Grad der Umsetzung von CI/CD ist hierbei entscheidend. Oftmals wird zunächst nur der Bereich Continuous Integration umgesetzt. Die Pipeline kann jedoch bis zum Kunden durchgezogen werden. Siehe dazu 1.1.

Letztendlich soll das CI/CD die vollständige Kontrolle über die zu entwickelnde oder anzupassende Anwendung behalten, vom Entwickler bis zum Kunden. Damit soll die Entwicklung der Anwendung sichergestellt werden.

Eine Pipeline stellt virtualisiert dar, welche Schritte nacheinander ausgeführt werden müssen, um den Weg der Anwendung einzuhalten.

## 2. Hudson/Jenkins

### 2.1. Jenkins als Tool für CI

Hudson wurde von Sun Microsystems, später Oracle entwickelt. In den 2000ern wurde ein Fork der Java-basierten Anwendung erstellt, das unter dem Namen Jenkins immer größere Popularität gewann.

Beide Anwendungen hatten und haben dasselbe Ziel: kontinuierliche Integrität. Als Open Source Anwendung existieren viele Alternativen, dennoch konnte sich Jenkins durchsetzen. Jenkins ist eine vollständige Webapplikation, die als Server-Dienst läuft. Standardmäßig auf Port 8080 des localhost. Die einfache Installation ist sicher ein weiterer Grund für die Popularität. Beispielsweise kann sie auf Mac, Linux und Windows, sowie auf Docker installiert werden. Die Jobs sind nicht vom Master abhängig, der Master stellt den Server dar. Die hohe Anzahl an Plugins ermöglicht eine flexible Einrichtung an die eigenen Bedürfnisse.

### 2.2. Installation

Die Installation wurde auf einem Windows-Computer durchgeführt. Unter Linux und Mac stehen mehr kommandozeilenbasierte Anwendungen zur Verfügung.


Download und Installieren der .exe

<https://www.jenkins.io/download/>

In den Diensten sollte nachgeschaut werden, dass der „jenkins“-Dienst läuft.

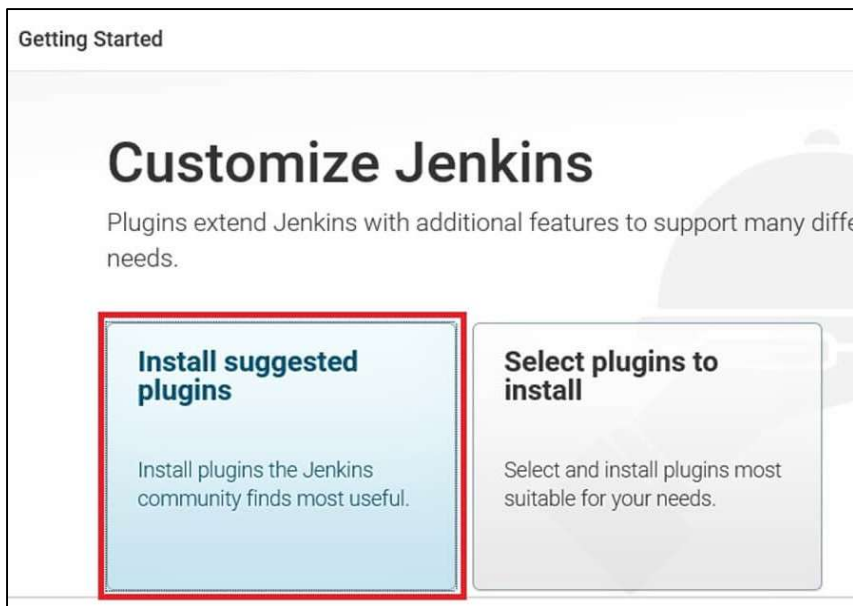
Die Anwendung startet nach dem Fertigstellen der Installation oder muss manuell im Browser der eigenen Wahl gestartet werden: <http://localhost:8080/>

Nun muss der Admin-Account mit dem Administrator Passwort bestätigt werden.

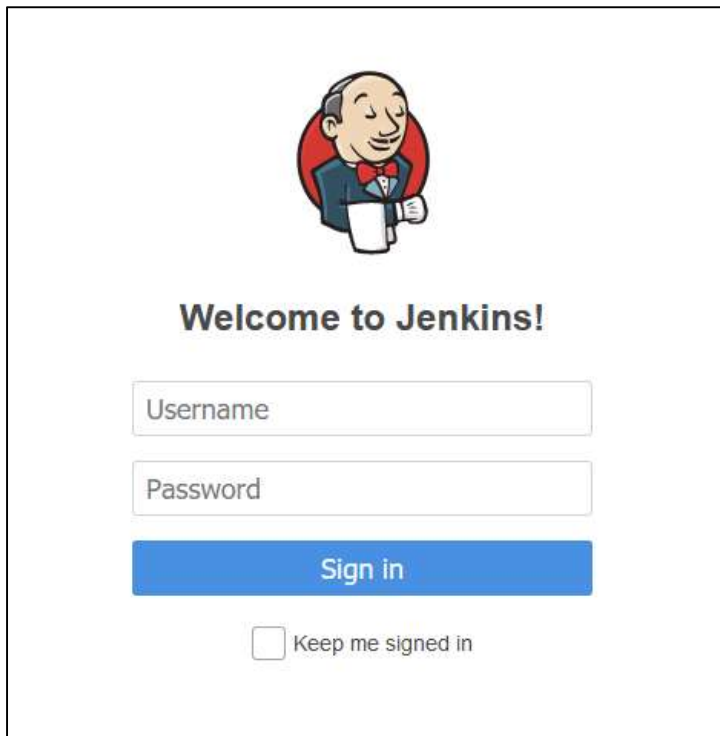
The image shows the 'Unlock Jenkins' screen. At the top, the title 'Unlock Jenkins' is displayed in a large, bold, black font. Below the title, a paragraph of text explains that a password has been written to the log and a file on the server. The file path 'C:\Program Files (x86)\Jenkins\secrets\initialAdminPassword' is highlighted in red. Below this, a prompt asks the user to copy the password from either location and paste it below. There is a text input field labeled 'Administrator password' with a light gray border. In the background, there is a faint, stylized illustration of a person wearing a hard hat and holding a tool.

Der rot hinterlegte Pfad gibt den Pfad in der lokalen Dateistruktur an, in der die angegebene Datei das Passwort für den Admin-Account enthält. Dieses Passwort muss nun in das Feld eingegeben werden.

Nach dem Installieren von den vorgeschlagenen Plugins...



...kann zunächst „[...]as admin“ fortgefahren werden.



Dazu das zuvor verwendete Admin-Passwort und den Usernamen „admin“ verwenden.

Das initiale Fenster in Jenkins sollte folgendermaßen aussehen:



Damit ist die Installation abgeschlossen.

### 2.3. Einen Job für lokale Dateien erstellen

1. Um einen Job zu erstellen, muss am linken Bildschirmrand auf „New Item“ geklickt werden.

Enter an item name

LocalItem

» Required field

**Freestyle project**

This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.

**Pipeline**

Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.

**Multi-configuration project**

Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.

**Bitbucket Team/Project**

Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defined markers.

**Folder**

Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates a separate namespace, so you can have multiple things of the same name as long as they are in different folders.

**GitHub Organization**

Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**

Creates a set of Pipeline projects according to detected branches in one SCM repository.

**Multibranch Pipeline with defaults**

Extend Multibranch pipeline plugin and build branches it the default preped pipeline script. Creates a set of Pipeline projects according to detected branches in one SCM repository.

Ein Name muss angegeben werden, hier beispielsweise „LocalItem“. Dann „Freestyle project“ anklicken und mit „OK“ bestätigen.

2. Informationen zum Job ausfüllen.

Eine Beschreibung unter „Description“ ist alternativ. Dann nichts verändern, außer „Build Triggers“ „Build periodically“ mit einem Haken markieren und „\* \* \* \* \*“ eingeben, um jede Minute einen Build erstellen zu lassen.

**Build Triggers**

☐ Trigger builds remotely (e.g., from scripts)

☒ Build periodically

Schedule

⚠ Do you really mean "every minute" when you say "\* \* \* \* \*"? Perhaps you meant "H \* \* \* \* \*" to poll once per hour  
Would last have run at Mittwoch, 20. Mai 2020 01:03 Uhr MESZ; would next run at Mittwoch, 20. Mai 2020 01:03 Uhr MESZ.

☐ GitHub hook trigger for GITScm polling

☐ Poll SCM

☐ Build after other projects are built

Unter „Build“ auf „Add build step“ klicken und das „Execute Shell“ anwählen.

**Build**

**Execute shell**

Command

[See the list of available environment variables](#)

Advanced...

Add build step ▼

Hier folgendes eintragen:

```
cd C:\Users\Labor\Documents\GitHub\e-portfolio-jenkins\myrogram\  
javac myProgram.class  
java myProgram
```

Mit Klick auf „Save“ wird der Job gespeichert. Nun kann der Job im Workspace folgendermaßen gesehen werden.

		LocalItem	N/A	N/A	N/A	
--	--	-----------	-----	-----	-----	--

Nach einer Weile sollte sich der Status des Jobs folgendermaßen geändert haben:

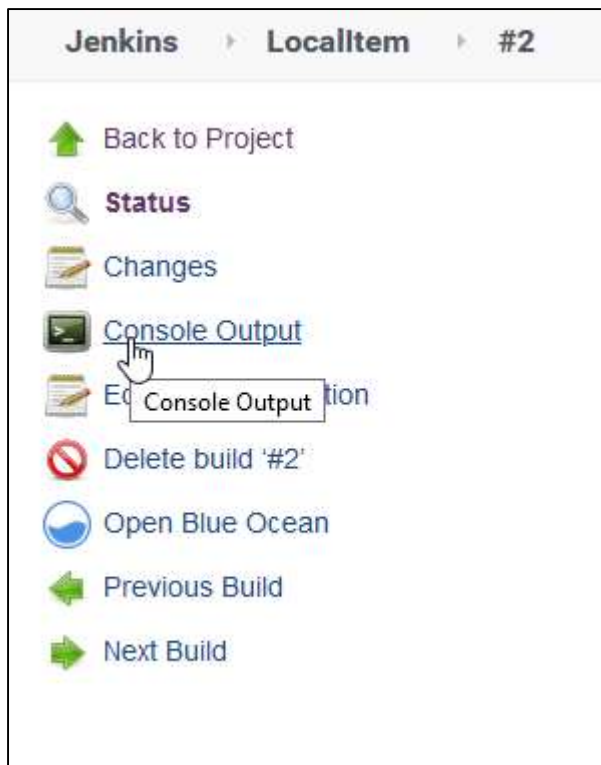
		LocalItem	16 sec - #1	N/A	0,23 sec	
--	--	-----------	-------------	-----	----------	--

Ist der Job nicht blau, sollte zunächst im Job selbst links unten der fehlgeschlagene Test geöffnet werden. Im Bild unten blau.



Anschließend kann der „Console Output“ angeschaut werden. Dieser listet die aufgetretenen Fehler.





## 2.4. Einen Job für remote Quellen auf GitHub erstellen

Hierfür bitte zunächst den Schritt 1 des vorherigen Unterkapitels ausführen, dann:

### 1. Informationen zum Job ausfüllen

**Source Code Management**

☐ None  
☒ Git

**Repositories**

Repository URL

**Please enter Git repository.**

Credentials

**Branches to build**

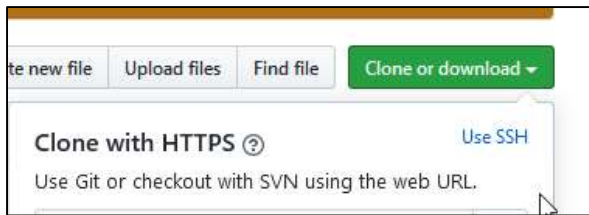
Branch Specifier (blank for 'any')

Repository browser

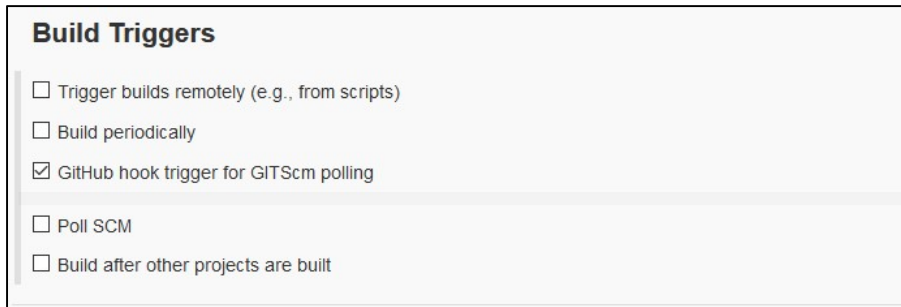
Additional Behaviours

☐ Mercurial  
☐ Subversion

Ein GitHub-Repository wird verlangt. Dazu bitte den Link des entsprechenden Repositories aus GitHub kopieren und einfügen:



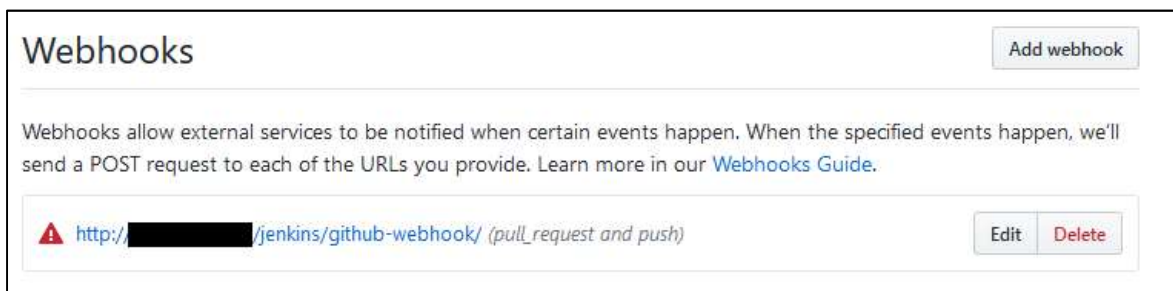
„Build Triggers“ zu GitHub einstellen:



Und auch, wie im vorherigen Unterkapitel beschrieben, die Execute Shell im Absatz „Build“ aufrufen und folgendes eintragen:

```
cd C:\Users\Labor\Documents\GitHub\e-portfolio-jenkins\myrogram\  
javac myProgram.class  
java myProgram
```

Mit Klick auf „Save“ ist zwar der Job erstellt, jedoch fehlt noch die Integration von GitHub, der Trigger für den Jenkins-Job. Dazu im GitHub-Repository in den Settings „Webhooks“ aufrufen und einen neuen Webhook erstellen. Dabei muss der Webhook folgendermaßen aussehen:



Im schwarzen Kasten steht dann die IPv4-Adresse des Jenkins-Servers. Wichtig ist die Endung „/jenkins/github-webhook/“!

Und auch hier, nach einem „Build now“ oder einer Änderung im angegebenen Repository kann folgendes beobachtet werden:



Der Job ist erfolgreich beendet worden.

## 2.5. Plugins und Tipps

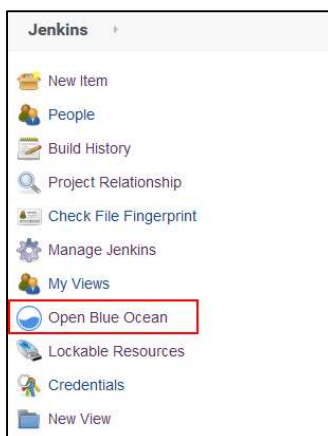
Green Balls, veränderte Anzeige des Job-Status für rot-grün-Sehswache.  
Email-Plugin, erweiterte Möglichkeiten zur Erstellung von Mail-Benachrichtigungen.  
Timestamp, erstellt auf der Konsolenausgabe einen Zeitstempel für die Jobs.  
CI-Game-Plugin, vergibt Punkte für erfolgreiche und nimmt Punkte für fehlerhafte Jobs.  
Blue Ocean, erleichtert das Erstellen von Pipelines.

Kann mit Docker, Git, Jira und vielem mehr verbunden werden. Slaves enthalten den Quellcode. Slaves sollten einfach auswechselbar sein, im Falle eines Slave-Ausfalls. Ein Master ist ein Jenkins-Server. Einer alleine sollte nicht alle Jobs ausführen. Mehrere Master sind in einer größeren Umgebung zu empfehlen.

## 2.6. Eine Pipeline mit Blue Ocean erstellen

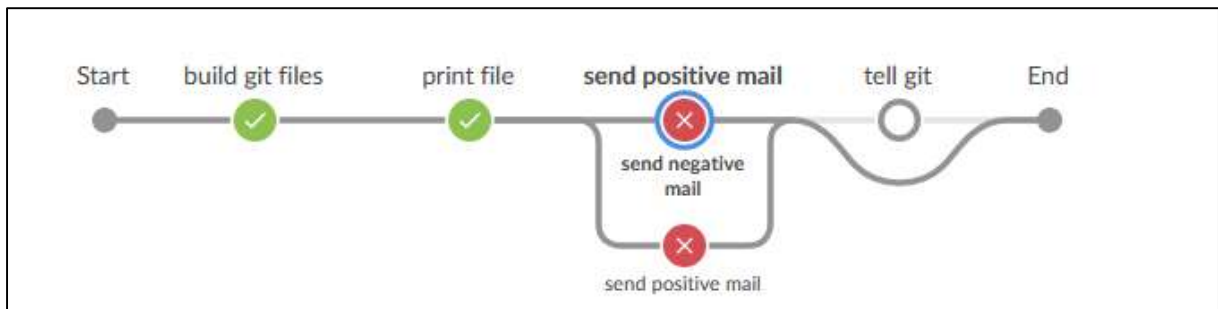
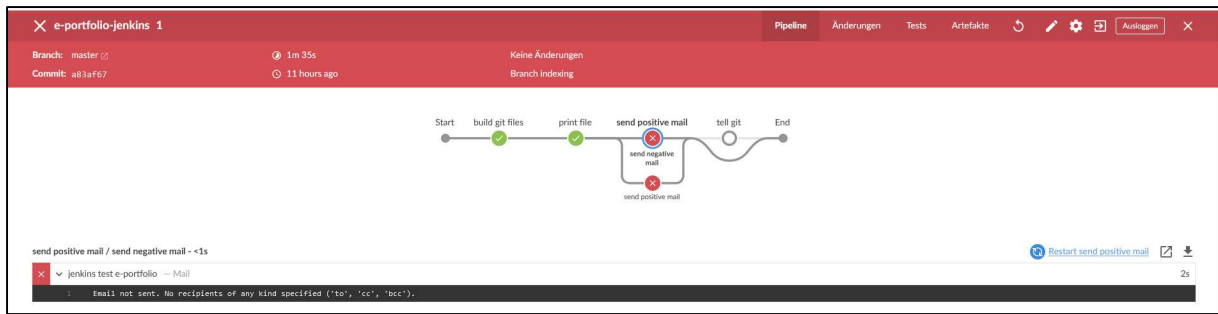
Über „Manage Jenkins“ zu „Manage Plugins“ zum Reiter „Available“ navigieren. Rechts oben „Blue Ocean“ eintippen und den ersten Eintrag installieren lassen.

Danach sollte die rechte Leisten-Ansicht folgendermaßen aussehen:



Mit Klick auf diesen „Open Blue Ocean“ kann über „Neue Pipeline“ eine neue Pipeline erstellt werden. Dazu die zutreffenden Eigenschaften der Pipeline im geführten Modus markieren und speichern. Über „Add Item“ können Jobs hinzugefügt an beinahe jeden beliebigen Ort hinzugefügt werden.

Die Pipeline ausführen lassen. In der folgenden Abbildung ist der Build und die Ausgabe eines einfachen Text-Dokuments erfolgreich ausgeführt worden. Jedoch konnte keine Mail gesendet werden. Dadurch wurden die weiteren Punkte „tell git“ und „End“ nicht ausgeführt werden.



Damit weiß der Betreuer des Projekts, dass an der Stelle „send positive mail“ ein Fehler passiert ist und hat eine Anhaltspunkt, wo sein Projekt steht.