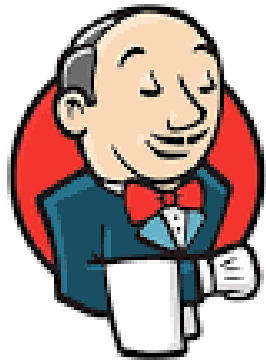


# Jenkins

Ein Seminar für ITZ Bund



# Jenkins

- Dies ist ein praktisches Seminar
  - Übungen vertiefen die vermittelten Inhalte
  - Ergänzend Diskussion, Demonstrationen
- Konventionen
  - Befehle werden in `Courier-Schriftart` dargestellt
  - Dateinamen werden in *`kursiver Courier-Schriftart`* dargestellt
  - Links werden in `unterstrichener Courier-Schriftart` dargestellt
- Zeitplan
  - 8 Unterrichtsblöcke mit jeweils etwa 90 Minuten
  - Durchschnittlich 6 Stunden Netto Seminarzeit

© Javacream

Javacream

Dr. Rainer Sawitzki

Alois-Gilg-Weg 6

81373 München

eMail: [training@rainer-sawitzki.de](mailto:training@rainer-sawitzki.de)

**Alle Rechte, einschließlich derjenigen des auszugsweisen Abdrucks, der fotomechanischen und elektronischen Wiedergabe vorbehalten.**

Grundlagen des Build-Managements	6
Jenkins-Übersicht	30
Jenkins Pipelines	40
Administration	47

1

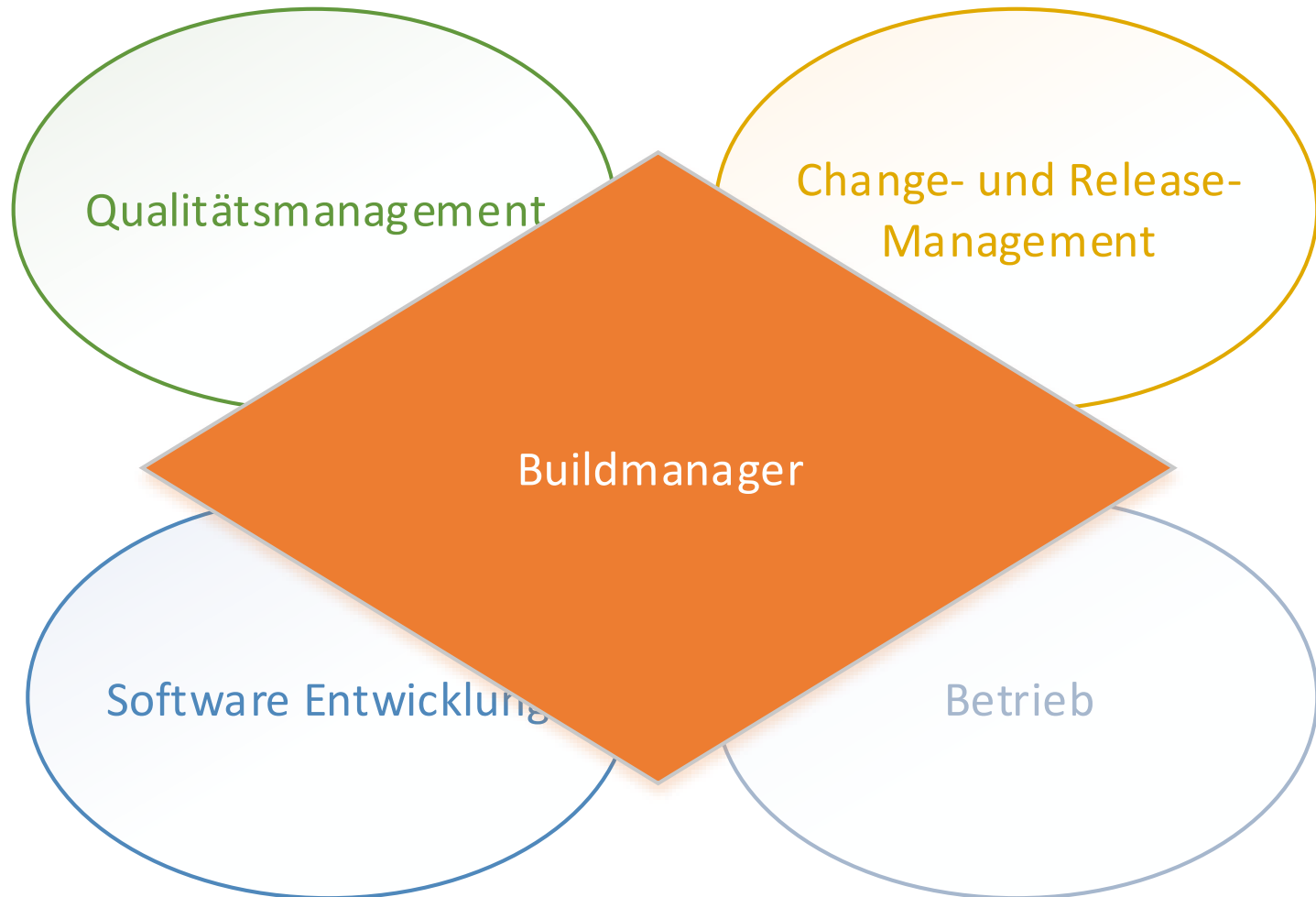
# GRUNDLAGEN DES BUILD-MANAGEMENTS

1.1

## **BUILDMANAGEMENT**

- Unter Buildmanagement verstehen wir alles, was dazu nötig ist aus einer Reihe von Werken ein oder mehrere Artefakte zu erzeugen. Dieser Vorgang soll dabei
  - automatisch
  - korrekt
  - und mit minimalem Aufwand
  - erfolgen
- Buildmanagement
  - umfasst dabei Werkzeuge, Infrastruktur sowie Prozesse und Verfahren
  - und bildet eine Schnittmenge zu den verwandten Gebieten Release-Management und Qualitäts-Management





**Paketieren und  
Ausliefern von  
(Release-)Versionen**

**Konfigurieren der  
Zielumgebungen**

**Deployen der  
Applikation auf den  
Zielsystemen**

**Pflege und  
Paketierung von  
Datenbank-Skripten**

**Einbeziehen des  
Betriebes zur  
Entwicklungszeit**

**Agile Prozesse**

**Virtualisierungs- und  
Cloudbasierte  
Ansätze**

**Automatisierung von  
Datacentern und  
Konfigmanagement**

1.2

## **ANFORDERUNGEN AN DEN BUILD- PROZESS**

- Nach dem Start des Buildprozesses ist kein Eingreifen mehr erforderlich.
  - Anti-Beispiel
    - "Nach dem Kompilieren kopieren Sie die Datei y in das Verzeichnis z und starten dann den Packager."

- Der Buildvorgang kann jederzeit wieder gestartet werden
  - (auch nach Jahren!
  - Das Ergebnis muss identisch sein

- Informationen über jeden Build-Vorgang
  - Erfolgreich / Fehlgeschlagen
  - Gründe für den Fehlschlag
  - Außerdem
    - Ergebnisse der Unit-Tests
    - Metriken
      - Code-Qualität
      - Test-Abdeckung
    - Statische Code Analysen

- Der Buildprozess muss automatisiert startbar sein
  - Start über die Kommandozeile
    - Gegenbeispiel: IDE-Builds

- Keine harten Abhängigkeiten
  - Environment
  - IP-Adressen
  - Dateipfade
- Im Idealfall sollten Build-Prozesse auf jedem beliebigen Rechner ausgeführt werden können
  - In der Praxis ist dies jedoch nicht immer möglich
    - iOS-Entwicklung nur auf MAC
    - .NET
  - Damit muss bei Bedarf der Build an einen geeigneten Rechner delegiert werden
- Gegenbeispiel
  - Die Magic Build-Machine



1.3

## **BUILD-WERKZEUGE**

- Der Buildvorgang wird programmiert
  - Shell-Skript
  - Make
  - Ant
  - Gradle

- Das Buildwerkzeug selbst definiert einen prototypischen Buildvorgang
- Jedes Projekt konfiguriert nur bestimmte Punkte:
  - Attribute (Projektname, Ergebnistyp)
  - Abweichungen vom Standard (Verzeichnisse)
  - Definierte Einsprungpunkte
- Beispiele
  - Apache Maven
  - Gradle

1.4

## **BEGRIFFE**

- Local (LOC)
- CI
- Team
- Development (DEV)
- Integration (INT)
- Approval (APP)
- Production (PROD)

## CI-Build

- Wir nach jeder Änderung am Sourcecode gebaut. 3-5min

## Long CI Build

- Führt Tests aus, die zu lange für den regulären CI-Build dauern

## Quality CI

- Überprüft im Rahmen des CIs die einheitlichen von Qualitätsregeln

## Trigger Build

- Dient zum kontrollierten Starten von anderen Jobs

## Nightly Build

- Läuft jede Nacht „auf der grünen Wiese“

## Quality Build

- z.B. SonarQube. Erstellt Qualitätsmetriken

## Deploy Jobs

- Deployt auf eine Zielumgebung

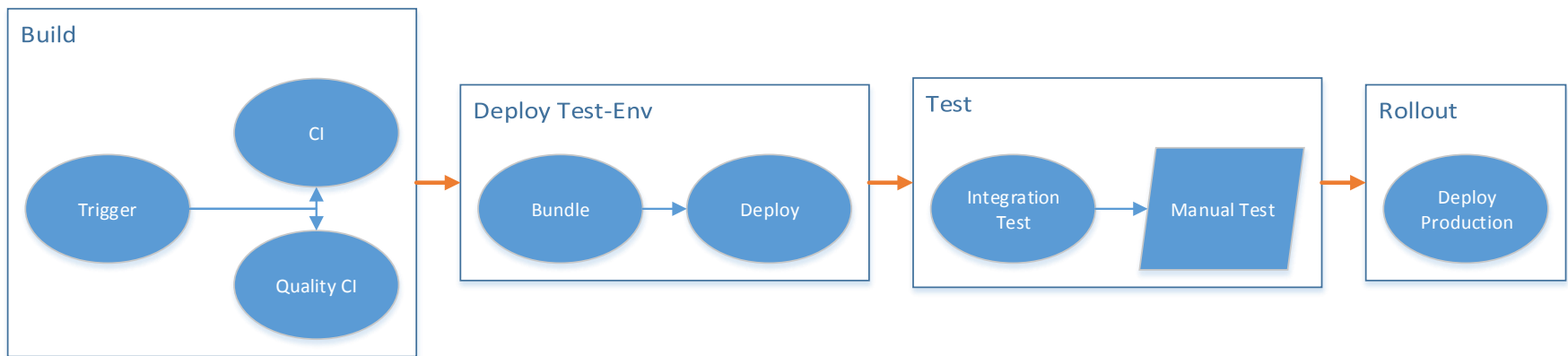
## Release Jobs

- Erzeugt einen Release, startet eine Release Pipeline

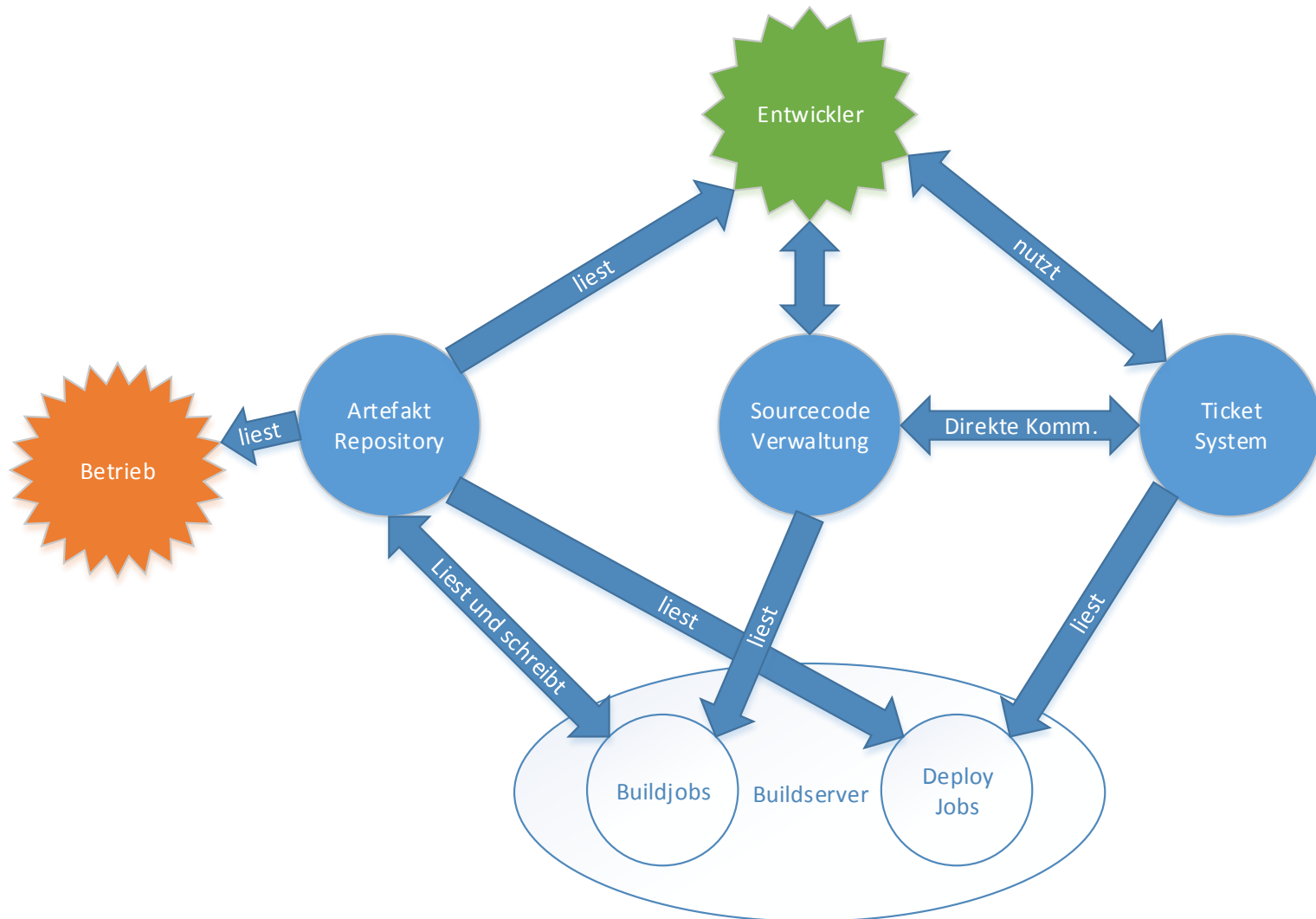
## Hilfsjobs

- Alles, was in keine andere Kategorie passt

- Build-Job
  - Trigger
  - Sourcecode-Repository
  - Buildumgebung
  - Build-Steps
  - Post-Processing
- Pipeline
  - Definition der notwendigen Schritte, um einen Build durchzuführen
- Buildknoten
  - Der Rechner, welcher einen Build-Prozess ausführt



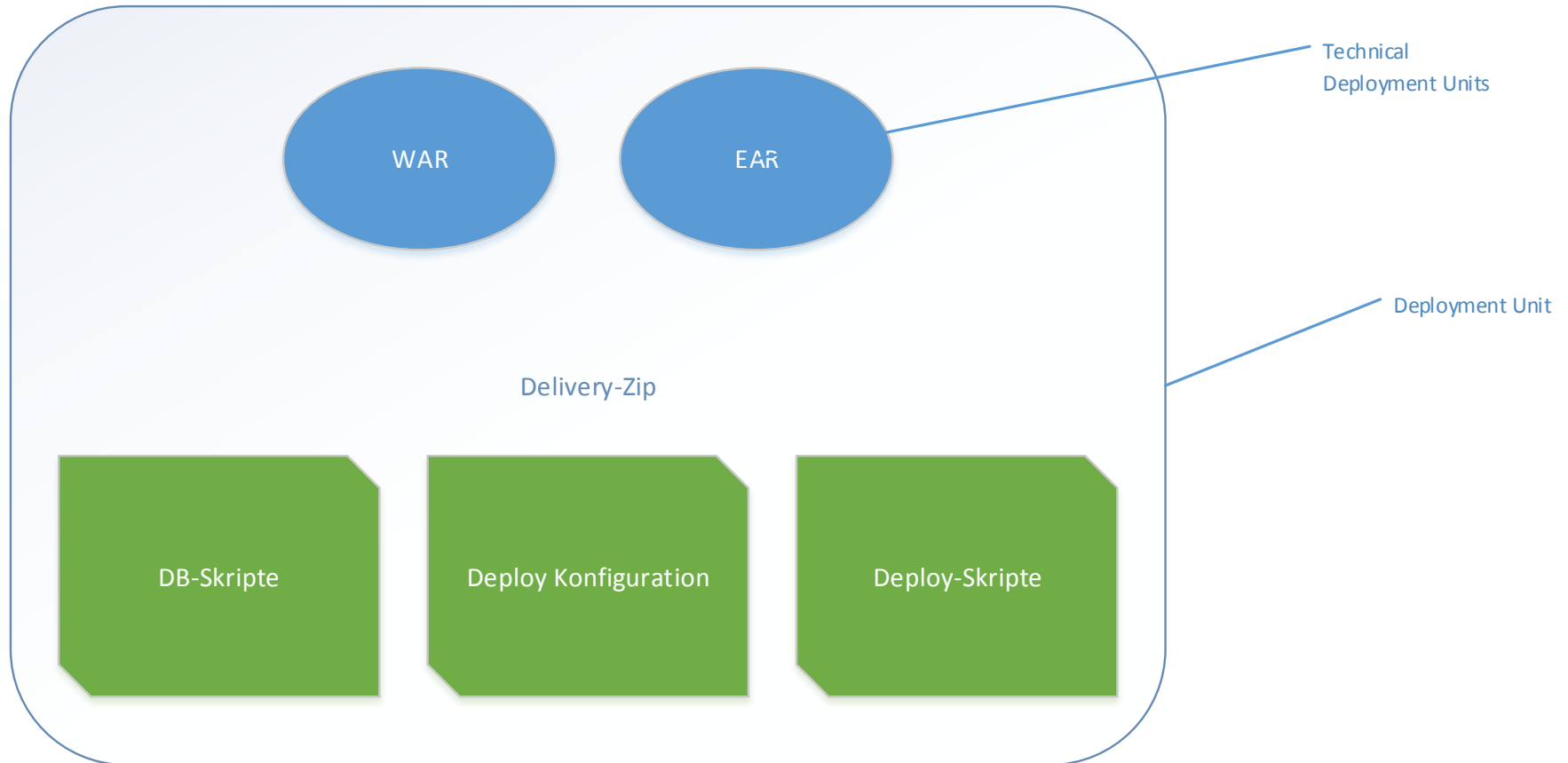




1.5

## DEPLOYMENT

- Continuous Delivery
- Continuous Deployment
- Deployment Unit (DU)
- Technical Deployment Unit (TDU)
- Stage
- Pipeline



- **Technologie-spezifische Mittel**
  - Tomcat Manager Applikation
  - Websphere wsadmin Kommando
- **Skripte**
  - Entsprechen den Buildwerkzeugen des Buildes
  - Kapseln die technologie-spezifischen Mittel
- **Provisioning**
  - Werkzeuge zum Aufsetzen einer Infrastruktur übernehmen jetzt auch das Deployment
  - In der Regel keine Anweisungen (deploye X), sondern Zielzustände (X muss in der Version 1.0 installiert sein)
  - Beispiel: Puppet, Chef, Ansible
- **Container**
  - Deployed wird nicht die Applikation, sondern die gesamte Infrastruktur. Ergebnis des Builds ist eine komplette Quasi-VM

2

## JENKINS-ÜBERSICHT

2.1

## **DOKUMENTATION**

## Guided Tour

- [Index](#)
- [Create your first Pipeline](#)
- [Running multiple steps](#)
- [Defining execution environments](#)
- [Using environment variables](#)
- [Recording test results and artifacts](#)
- [Cleaning up and notifications](#)
- [Deployment](#)

## User Handbook (PDF)

- [Getting Started with Jenkins](#)
- [Using Jenkins](#)
- [Managing Jenkins](#)
- [Best Practices](#)
- [Pipeline](#)
- [Blue Ocean](#)
- [Jenkins Use-Cases](#)
- [Operating Jenkins](#)
- [Scaling Jenkins](#)
- [Appendix](#)
- [Glossary](#)

## Resources

- [Pipeline Syntax reference](#)
- [Pipeline Steps reference](#)
- [LTS Upgrade Guide](#)

## Recent Tutorials

- [Pipeline Development Tools](#)
- [Getting Started with the Blue Ocean Dashboard](#)
- [Getting Started with Blue Ocean's Activity View](#)

[View all tutorials](#)

## Jenkins Documentation

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks such as building, testing, and deploying software. Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with the Java Runtime Environment installed.

## Guided Tour

This guided tour will use the "standalone" Jenkins distribution which requires a minimum of Java 7, though Java 8 is recommended. A system with more than 512MB of RAM is also recommended.

1. [Download Jenkins](#).
2. Open up a terminal in the download directory and run `java -jar jenkins.war`
3. Browse to `http://localhost:8080` and follow the instructions to complete the installation.
4. Many Pipeline examples require an [installed Docker](#) on the same computer as Jenkins.

When the installation is complete, start putting Jenkins to work and create a [Pipeline](#).

Jenkins Pipeline is a suite of plugins which supports implementing and integrating continuous delivery pipelines into Jenkins. Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code".

A [Jenkinsfile](#) is a text file that contains the definition of a Jenkins Pipeline and is checked into source control. <sup>[1]</sup> This is the foundation of "Pipeline-as-Code"; treating the continuous delivery pipeline a part of the application to be version and reviewed like any other code. Creating a [Jenkinsfile](#) provides a number of immediate benefits:

- Automatically create Pipelines for all Branches and Pull Requests
- Code review/iteration on the Pipeline
- Audit trail for the Pipeline
- Single source of truth <sup>[2]</sup> for the Pipeline, which can be viewed and edited by multiple members of the project.

While the syntax for defining a Pipeline, either in the web UI or with a [Jenkinsfile](#), is the same, it's generally considered best practice to define the Pipeline in a [Jenkinsfile](#) and check that in to source control.

[Continue to "Create your first Pipeline"](#)

1. [https://en.wikipedia.org/wiki/Source\\_control\\_management](https://en.wikipedia.org/wiki/Source_control_management)
2. [https://en.wikipedia.org/wiki/Single\\_Source\\_of\\_Truth](https://en.wikipedia.org/wiki/Single_Source_of_Truth)



# Jenkins User Handbook

[jenkinsci-docs@googlegroups.com](mailto:jenkinsci-docs@googlegroups.com)

2.2

## INSTALLATION

- **Standalone**
  - mit integriertem Web Server
- **Als Web-Archiv**
  - notwendig ist dann noch ein Java-Applikationsserver
    - z.B. ein Apache Tomcat
- **Als Docker-Container**
  - Komplett-Pakete auf dem Docker-Hub
    - jenkinsci/blueocean

2.3

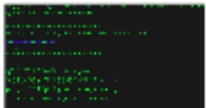
## ERSTES STARTEN

When you first access a new Jenkins instance, you are asked to unlock it using an automatically-generated password.

1. Browse to `localhost:8080` (or whichever port you configured for Jenkins when installing it) and wait until the **Unlock Jenkins** page appears.



2. From the Jenkins console log output, copy the automatically-generated alphanumeric password (between the 2 sets of asterisks).



3. On the **Unlock Jenkins** page, paste this password into the **Administrator password** field and click **Continue**.

#### Notes:

- If you ran Jenkins in Docker in detached mode, you can access the Jenkins console log from the Docker logs ([above](#)).
- The Jenkins console log indicates the location (in the Jenkins home directory) where this password can also be obtained. This password must be entered in the setup wizard on new Jenkins installations before you can access Jenkins's main UI. This password also serves as the default administrator account's password (with username "admin") if you happen to skip the subsequent user-creation step in the setup wizard.

After [unlocking Jenkins](#), the **Customize Jenkins** page appears. Here you can install any number of useful plugins as part of your initial setup.

Click one of the two options shown:

- **Select plugins to install** - to choose which set of plugins to initially install. When you first access the plugin selection page, the suggested plugins are selected by default.

## NOTE

If you are not sure what plugins you need, choose **Install suggested plugins**. You can install (or remove) additional Jenkins plugins at a later point in time via the [Manage Jenkins](#) > [Manage Plugins](#) page in Jenkins.

The setup wizard shows the progression of Jenkins being configured and your chosen set of Jenkins plugins being installed. This process may take a few minutes.

Finally, after [customizing Jenkins with plugins](#), Jenkins asks you to create your first administrator user.

1. When the **Create First Admin User** page appears, specify the details for your administrator user in the respective fields and click **Save and Finish**.
2. When the **Jenkins is ready** page appears, click **Start using Jenkins**.

**Notes:**

- This page may indicate **Jenkins is almost ready!** instead and if so, click **Restart**.
  - If the page does not automatically refresh after a minute, use your web browser to refresh the page manually.
3. If required, log in to Jenkins with the credentials of the user you just created and you are ready to start using Jenkins!

**IMPORTANT**

From this point on, the Jenkins UI is only accessible by providing valid username and password credentials.

3

## JENKINS PIPELINES



3.1

## GRUNDLAGEN

Jenkins Pipeline (or simply "Pipeline" with a capital "P") is a suite of plugins which supports implementing and integrating *continuous delivery pipelines* into Jenkins.

A *continuous delivery pipeline* is an automated expression of your process for getting software from version control right through to your users and customers. Every change to your software (committed in source control) goes through a complex process on its way to being released. This process involves building the software in a reliable and repeatable manner, as well as the progression of the built software (called a "build") through multiple stages of testing and deployment.

Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the [Pipeline Domain Specific Language \(DSL\) syntax](#). [1: [Domain-Specific Language](#)]

Pipeline provides an extensible set of tools for modeling simple-to-complex delivery pipelines "as code" via the [Pipeline Domain Specific Language \(DSL\) syntax](#). [1: [Domain-Specific Language](#)]

Typically, the definition of a Jenkins Pipeline is written into a text file (called a **Jenkinsfile**) which in turn is checked into a project's source control repository. [2: [Source Control Management](#)] This is the foundation of "Pipeline-as-Code"; treating the continuous delivery pipeline a part of the application to be versioned and reviewed like any other code. Creating a **Jenkinsfile** provides a number of immediate benefits:

- Automatically create Pipelines for all Branches and Pull Requests
- Code review/iteration on the Pipeline
- Audit trail for the Pipeline
- Single source of truth [3: [en.wikipedia.org/wiki/Single\\_Source\\_of\\_Truth](https://en.wikipedia.org/wiki/Single_Source_of_Truth)] for the Pipeline, which can be viewed and edited by multiple members of the project.

## Step

A single task; fundamentally steps tell Jenkins *what* to do. For example, to execute the shell command `make` use the `sh` step: `sh 'make'`. When a plugin extends the Pipeline DSL, that typically means the plugin has implemented a new *step*.

## Node

Most *work* a Pipeline performs is done in the context of one or more declared `node` steps. Confining the work inside of a node step does two things:

1. Schedules the steps contained within the block to run by adding an item to the Jenkins queue. As soon as an executor is free on a node, the steps will run.
2. Creates a workspace (a directory specific to that particular Pipeline) where work can be done on files checked out from source control.

### CAUTION

Depending on your Jenkins configuration, some workspaces may not get automatically cleaned up after a period of inactivity. See tickets and discussion linked from [JENKINS-2111](#) for more information.

## Stage

`stage` is a step for defining a conceptually distinct subset of the entire Pipeline, for example: "Build", "Test", and "Deploy", which is used by many plugins to visualize or present Jenkins Pipeline status/progress. [6: [Blue Ocean](#), [Pipeline Stage View plugin](#)]

3.2

## DETAILS

- Getting Started with Pipeline
- Jenkinsfile
- Jenkinsfile Syntax

4

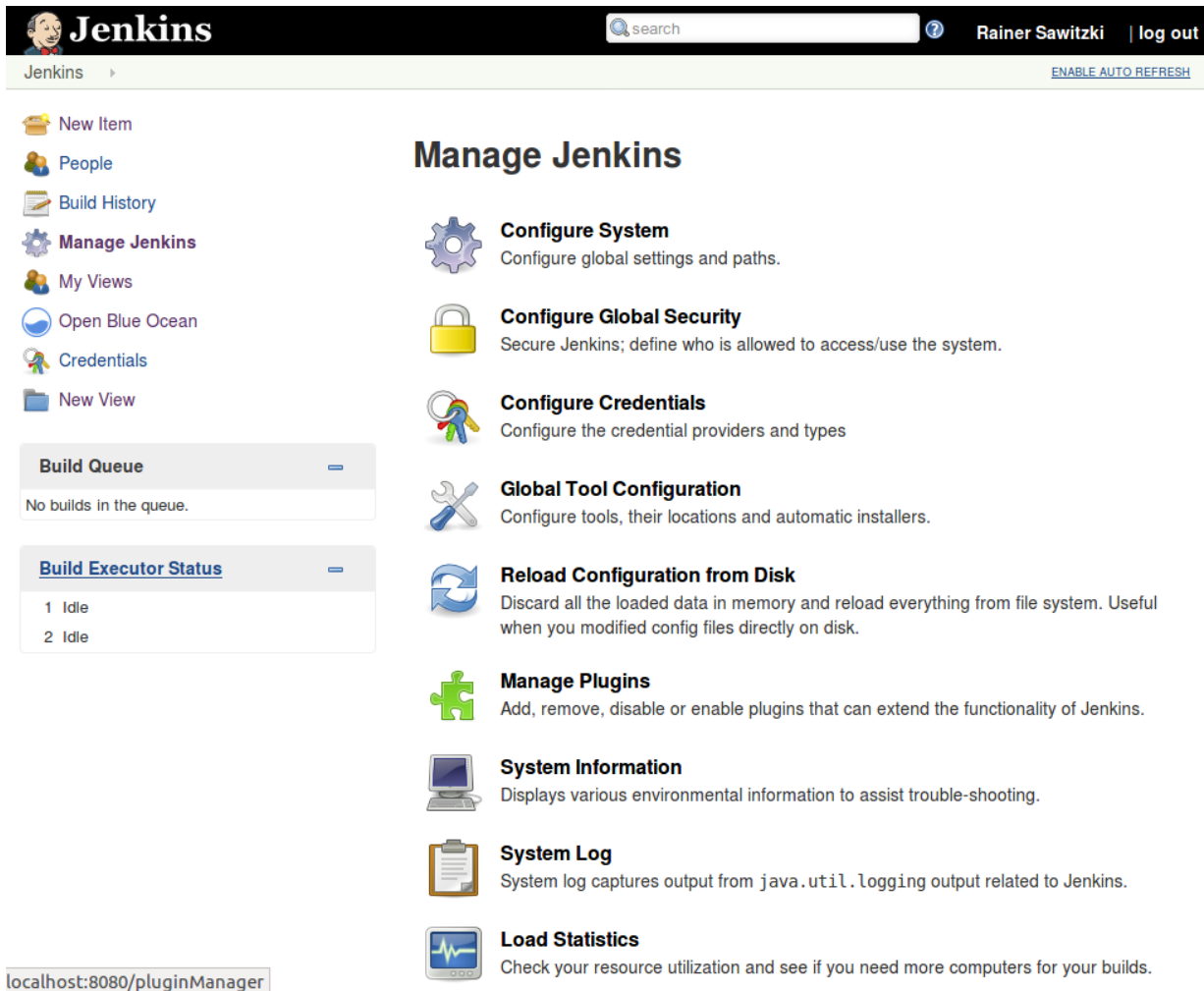
## ADMINISTRATION

4.1

## KONFIGURATION



- Nur für Administratoren sichtbar ist der Befehl „Manage Jenkins“



The screenshot shows the Jenkins web interface. At the top is a black header with the Jenkins logo, a search bar, and the user name 'Rainer Sawitzki' with a 'log out' link. Below the header is a green navigation bar with 'Jenkins' and a link to 'ENABLE AUTO REFRESH'. On the left is a sidebar with links: 'New Item', 'People', 'Build History', 'Manage Jenkins' (highlighted), 'My Views', 'Open Blue Ocean', 'Credentials', and 'New View'. Below the sidebar are two panels: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two 'Idle' executors). The main content area is titled 'Manage Jenkins' and contains a list of configuration options, each with an icon and a description:

- Configure System**: Configure global settings and paths.
- Configure Global Security**: Secure Jenkins; define who is allowed to access/use the system.
- Configure Credentials**: Configure the credential providers and types.
- Global Tool Configuration**: Configure tools, their locations and automatic installers.
- Reload Configuration from Disk**: Discard all the loaded data in memory and reload everything from file system. Useful when you modified config files directly on disk.
- Manage Plugins**: Add, remove, disable or enable plugins that can extend the functionality of Jenkins.
- System Information**: Displays various environmental information to assist trouble-shooting.
- System Log**: System log captures output from `java.util.logging` output related to Jenkins.
- Load Statistics**: Check your resource utilization and see if you need more computers for your builds.

At the bottom left, the address bar shows 'localhost:8080/pluginManager'.

- System-Konfiguration
- Security
- PlugIns-Installation
- Data Storage
- Master/Slave

- Mit Hilfe von PlugIns können in Jenkins weitere Funktionen integriert werden
  - Eigentlich ist Jenkins selbst „nur“ eine Laufzeitumgebung für PlugIns
- PlugIns werden von der jenkins.io zur Verfügung gestellt



4.2

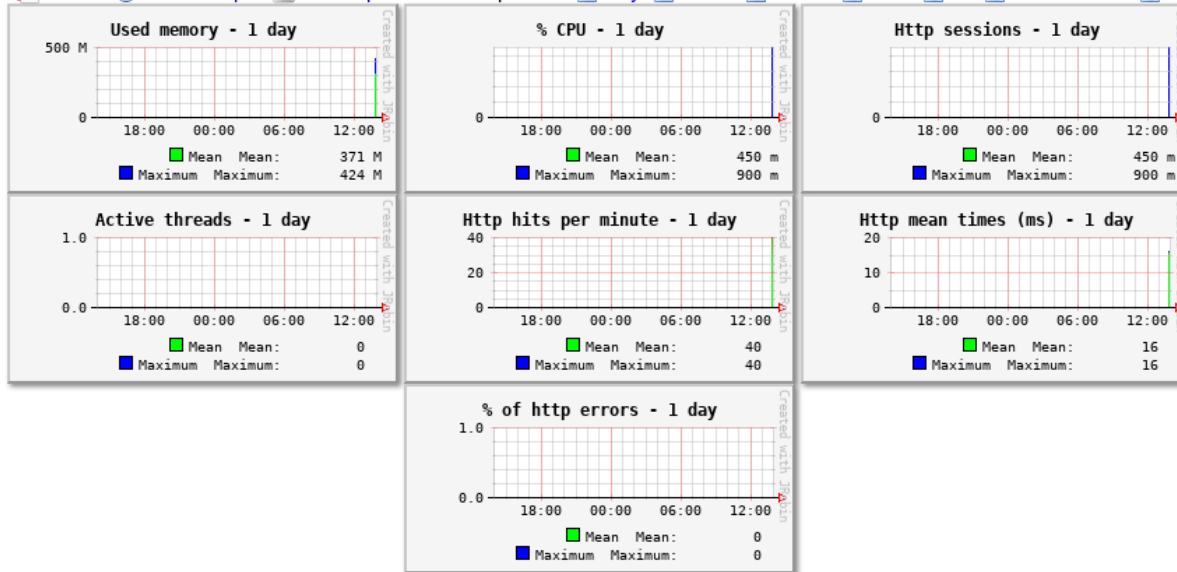
## ÜBERWACHUNG

- Jenkins ist ein Java-Prozess
  - damit greifen die Überwachungs-Werkzeuge der Java Virtual Machine
    - Java Management Extension, JMX
  - Speicher
  - Garbage Collection
- Jenkins läuft in einem Applikationsserver
  - Wahrscheinlich in einem Apache Tomcat
  - Überwachbar sind damit
    - Zugriffe
    - Sessions
    - Abbrüche und sonstige Fehler
- <https://wiki.jenkins.io/display/JENKINS/Monitoring>

## Statistics of JavaMelody monitoring taken at 08/07/18 13:54 on \_ad20c19ce890 (Jenkins v2.121.1, 2.121.1)

[Donate](#)

[Update](#)
[PDF](#)
[Online help](#)
[Desktop](#)
 Choice of period : 
 [Day](#)
[Week](#)
[Month](#)
[Year](#)
[All](#)
[Customized](#)
[By deployment](#)


[Other charts](#)

### Statistics http - 1 day

Request	% of cumulative time	Hits	Mean time (ms)	Max time (ms)	Standard deviation	% of cumulative cpu time	Mean cpu time (ms)	Mean allocated Kb	% of system error	Mean size (Kb)
http global	100	40	16	345	59	100	13	1,858	0.00	8
http warning	0	0	-1	0	-1	0	-1	-1	0.00	0
http severe	74	2	251	345	132	82	224	30,947	0.00	85

32 hits/min on 35 requests

[Details](#)

### Statistics http system errors - 1 day

None

### Statistics system errors logs - 1 day

4.3

## **MASTER/SLAVE**

- Skalierbarkeit
  - Build-Prozesse benötigen eine beträchtliche Rechnerleistung
- Environment
  - Manche Prozesse benötigen komplett unterschiedliche Build-Umgebungen
    - z.B. iOS



- Der Master stellt die gesamte Administrationsoberfläche zur Verfügung
  - und enthält damit die sämtliche Job-Definitionen
- Zusätzlich registriert der Master alle Slaves
  - Der Master startet den Jenkins-Prozess via SSH
  - oder die Jenkins-Instanzen werden händisch gestartet und registrieren sich beim Master
- Der Master verteilt die Prozess-Definitionen auf die Knoten
- Auf dem Master werden sämtliche Build-Ergebnisse gesammelt und zentral verfügbar gemacht