



DevOps Continuous Integration (2/2)



Building Competence. Crossing Borders.

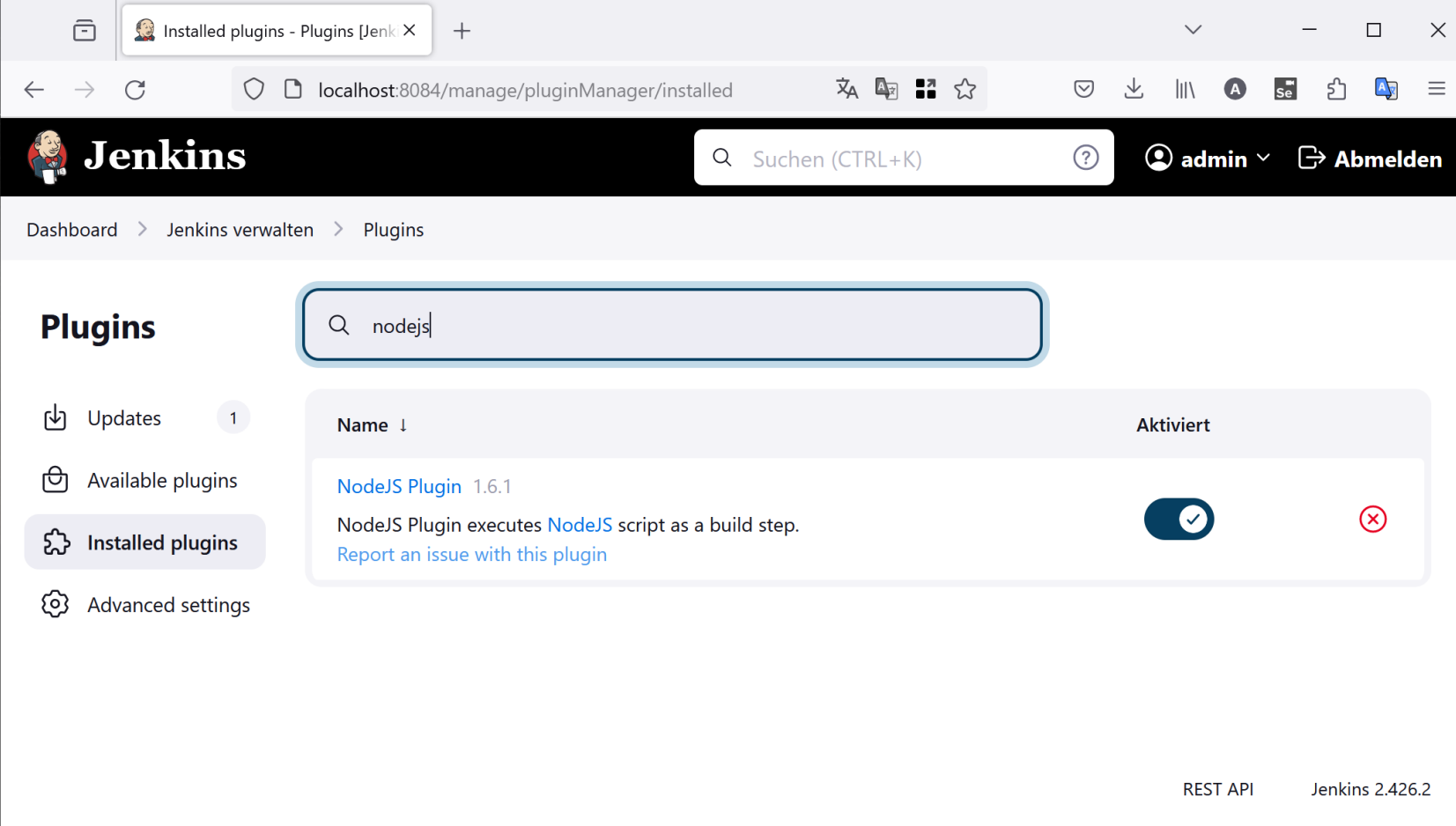
Jenkins und Node Tutorial

Bisher haben wir erst das Backend gebaut – das Frontend fehlt noch

NodeJS Build Step Plugin

Tutorial

Wir benötigen NodeJS, um das Frontend zu bauen. Es gibt ein Plugin für Jenkins (welches aber bereits vorinstalliert ist).



The screenshot shows the Jenkins web interface at localhost:8084. The top navigation bar includes the Jenkins logo, a search bar, and user information (admin). The breadcrumb trail is Dashboard > Jenkins verwalten > Plugins. The main section is titled 'Plugins' and features a search bar with 'nodejs' entered. On the left sidebar, 'Installed plugins' is selected. The main content area displays a table of installed plugins:

Name ↓	Aktiviert
NodeJS Plugin 1.6.1 NodeJS Plugin executes NodeJS script as a build step. Report an issue with this plugin	<input checked="" type="checkbox"/>

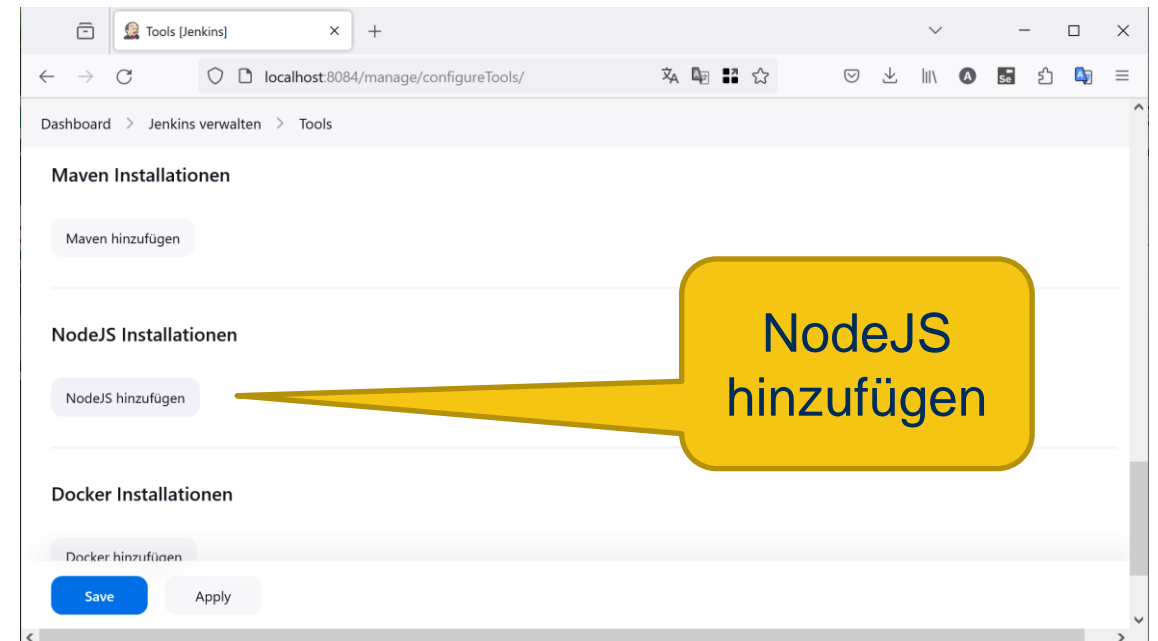
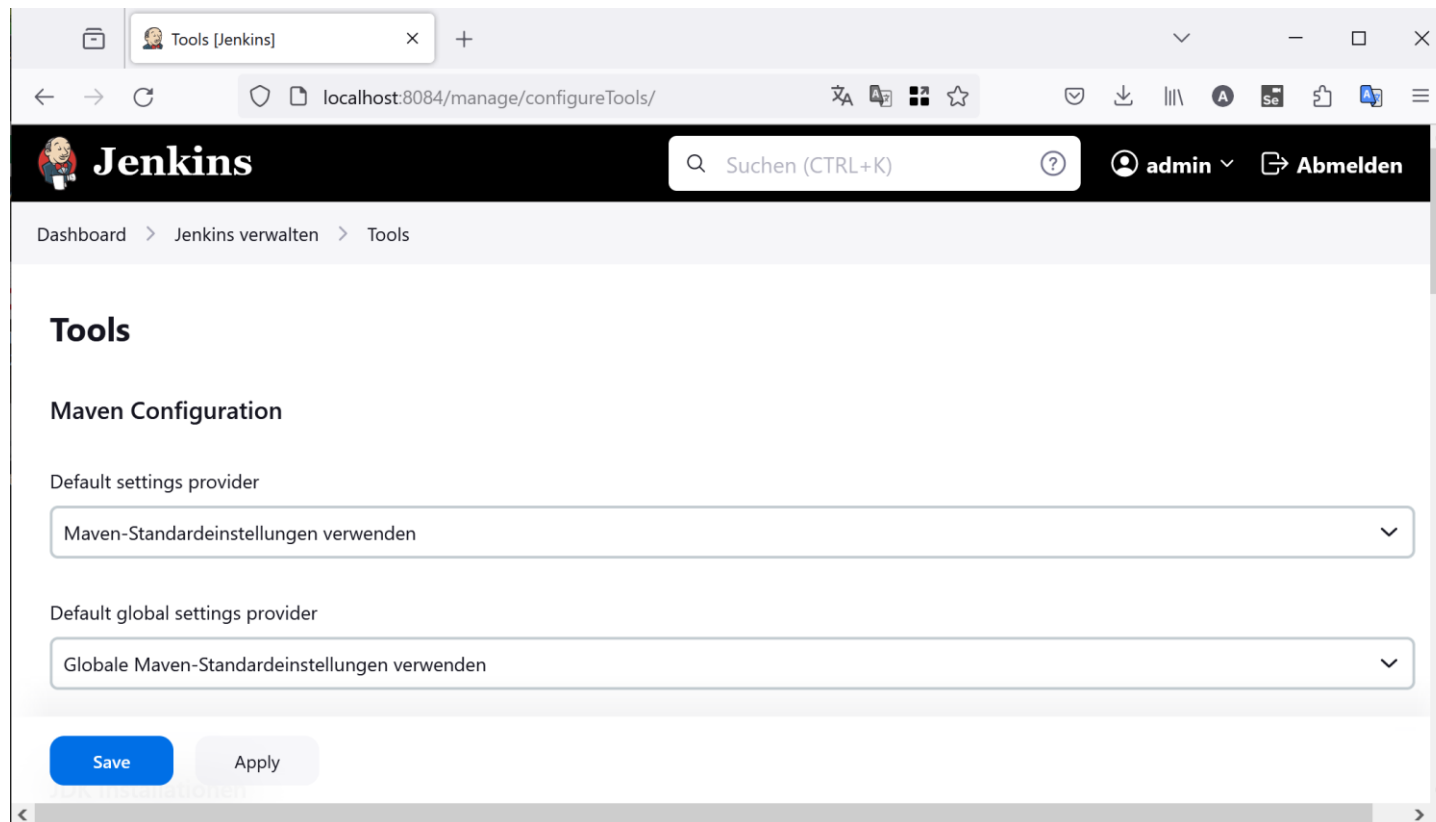
At the bottom right, it shows 'REST API' and 'Jenkins 2.426.2'.

Tools: NodeJS installieren

Tutorial

NodeJS Installation: Das NodeJS Plugin enthält aber noch keine NodeJS Installation!

Dashboard > Jenkins verwalten > Tools: Tools wie Java, NodeJS werden in Jenkins global verwaltet.



Global Tool Configuration: Viele Tools können auto-installiert werden, so auch NodeJS

The screenshot shows the Jenkins 'Tools [Jenkins]' configuration page for NodeJS. The form includes the following fields and options:

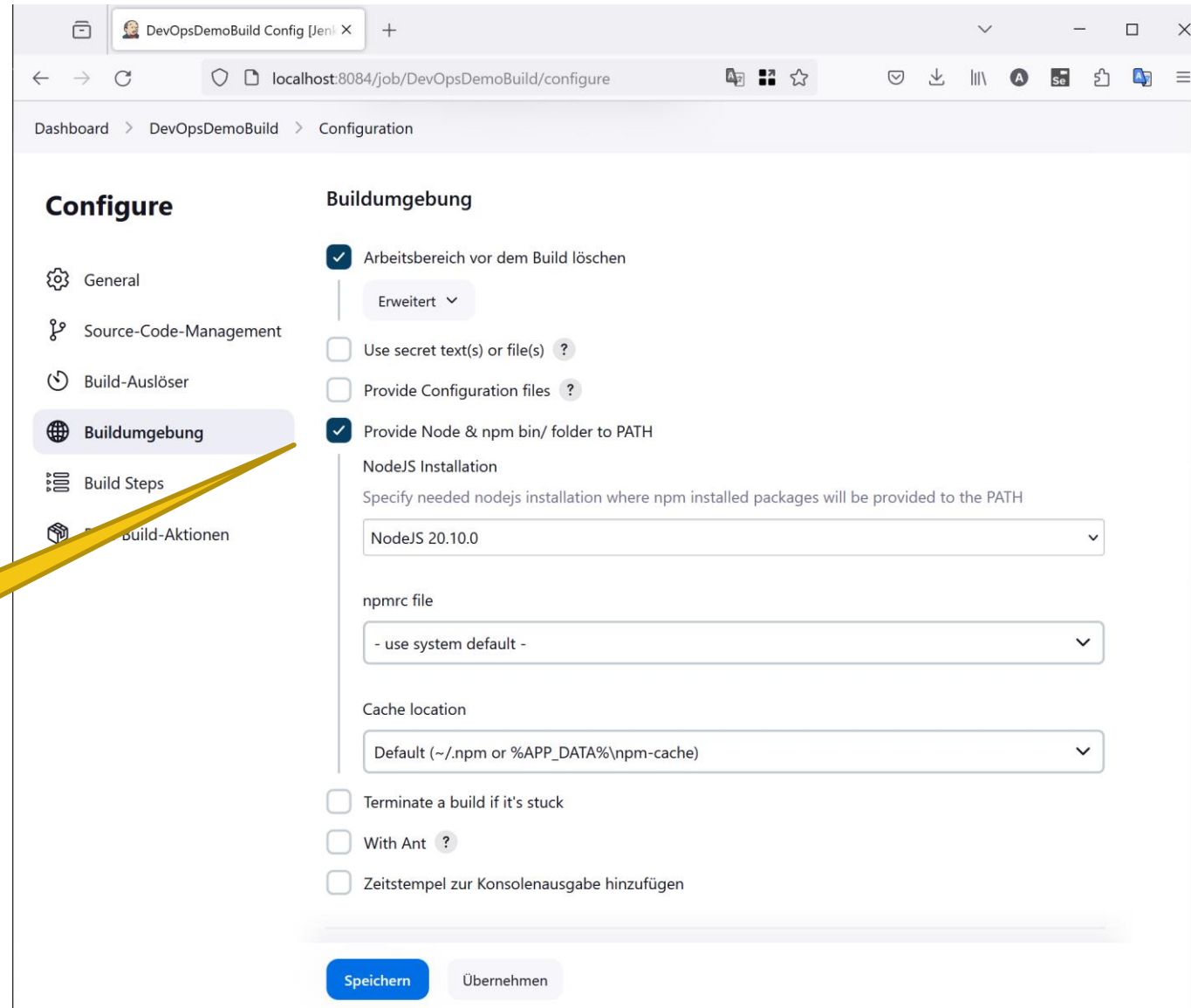
- Name:** A text input field containing 'NodeJS 20.10.0'.
- Automatisch installieren:** A checked checkbox with a help icon.
- Install from nodejs.org:** A section containing:
 - Version:** A dropdown menu showing 'NodeJS 20.10.0'.
 - Force 32bit architecture:** An unchecked checkbox.
 - Global npm packages to install:** A text input field with placeholder text: 'Specify list of packages to install globally -- see npm install -g. Note that you can fix the packages version by using the syntax `packageName@version`'.
 - Global npm packages refresh hours:** A text input field containing '72'.
- Buttons:** 'Save' and 'Apply' buttons at the bottom.

Yellow callout bubbles highlight the 'Name' field, the 'Version' dropdown, and the 'Save' button.

Add NodeJS Build Environment to Build

Tutorial

NodeJS
hinzufügen



Dashboard > DevOpsDemoBuild > Configuration

Configure

- General
- Source-Code-Management
- Build-Auslöser
- Buildumgebung**
- Build Steps
- Build-Aktionen

Buildumgebung

- ☒ Arbeitsbereich vor dem Build löschen
 - Erweitert ▾
- ☐ Use secret text(s) or file(s) ?
- ☐ Provide Configuration files ?
- ☒ Provide Node & npm bin/ folder to PATH
 - NodeJS Installation
 - Specify needed nodejs installation where npm installed packages will be provided to the PATH
 - NodeJS 20.10.0 ▾
 - npmrc file
 - use system default - ▾
 - Cache location
 - Default (~/.npm or %APP_DATA%\npm-cache) ▾
- ☐ Terminate a build if it's stuck
- ☐ With Ant ?
- ☐ Zeitstempel zur Konsolenausgabe hinzufügen

Speichern Übernehmen

Nun können die NodeJS Befehle bei Buildverfahren (Execute Shell) eingefügt werden:

```
npm install --prefix frontend  
npm run build --prefix frontend
```

Dashboard > DevOpsDemoBuild > Configuration

Configure

- General
- Source-Code-Management
- Build-Auslöser
- Buildumgebung
- Build Steps**
- Post-Build-Aktionen

Shell ausführen ?

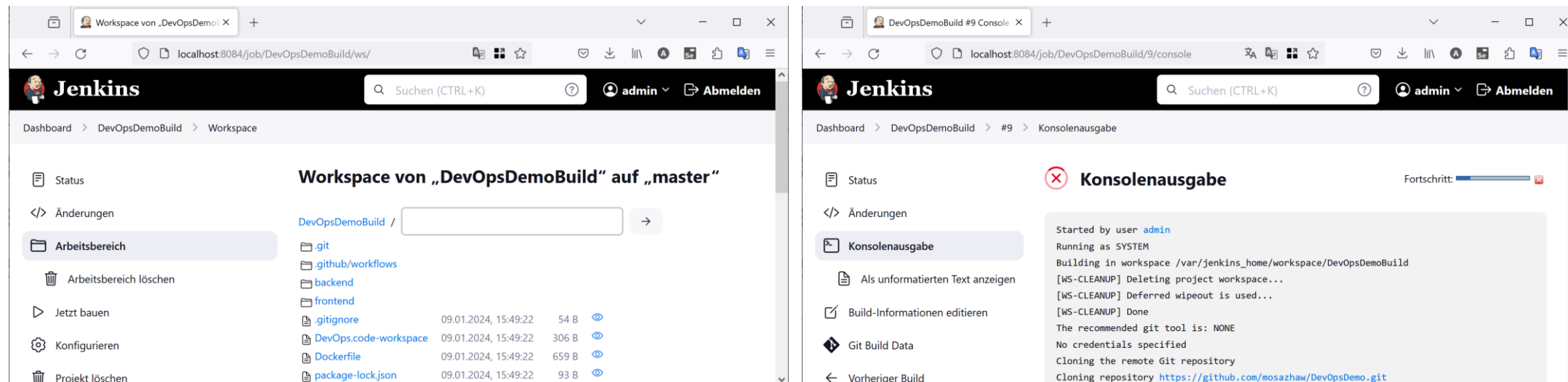
Befehl

[Liste der verfügbaren Umgebungsvariablen](#)

```
npm install --prefix frontend  
npm run build --prefix frontend
```

Erweitert ▾

Bei Problemen können der Log (**Console Output**) sowie alle Dateien (**Workspace**) analysiert werden.



```
chunk {main} main.js, main.js.map (main) 130 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 296 kB [initial] [rendered]
chunk {polyfills-es5} polyfills-es5.js, polyfills-es5.js.map (polyfills-es5) 679 kB [initial]
[rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 1.47 MB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 12.7 MB [initial] [rendered]
Date: 2020-06-26T14:29:02.959Z - Hash: 4add8373577a7fd17a89 - Time: 29252ms
```


Jenkins Pipelines

Jenkins Builds «programmieren»

Klassische **Jenkins**-Jobs, sogenannte **Freestyle** Jobs, werden über das Webinterface konfiguriert. «**Pipeline** as Code» bedeutet, dass die Jobs nicht mehr über das User Interface konfiguriert, sondern als Code hinterlegt werden.

Jenkins und Pipelines

Jenkins Pipelines

Jenkins Pipelines sind ein Set von Plugins, welche das Erstellen und Integrieren von Continuous Delivery (CD) Pipelines in Jenkins unterstützt.

CD Pipeline

Automatisierung des Prozesses vom Source Code (im Version Control System) bis zum fertigen Deployment für Benutzer und Kunden.

Jede Änderung am Source Code durchläuft (im Erfolgsfall) diesen Prozess.

Der Prozess umfasst das Zusammenbauen der Software sowie das Testen und Installieren der Software.

Deklarative Variante

Pipeline

Eine Pipeline-Definition umfasst alle Etappen (Stages) um eine Applikation zusammenzubauen (Build), zu testen (Test) und dann auszuliefern (Deploy).

Stage

Eine Stage ist eine Untergruppe von Schritten, welche innerhalb der Pipeline ausgeführt werden. (z.B. "Build", "Test" and "Deploy" stages).

Step

A single task.

```
pipeline {
    agent any
    stages {
        stage('Build') {
            steps {
                //
            }
        }
        stage('Test') {
            steps {
                //
            }
        }
        stage('Deploy') {
            steps {
                //
            }
        }
    }
}
```

Jenkins Pipeline Tutorial

Erstellen einer Pipeline in Jenkins

Neue Pipeline in Jenkins erstellen

Tutorial

Element anlegen [Jenkins] x +

localhost:8084/view/all/newJob

Jenkins Suchen (CTRL+K) admin Abmelden

Dashboard > Alle >

Geben Sie einen Element-Namen an

DevOpsPipeline

» Pflichtfeld

"Free Style"-Softwareprojekt bauen
Dieses Profil ist das meistgenutzte in Jenkins. Jenkins baut Ihr Projekt, wobei Sie universell jedes SCM System mit jedem Build-Verfahren kombinieren können. Dieses Profil ist nicht nur auf das Bauen von Software beschränkt, sondern kann darüber hinaus auch für weitere Anwendungsgebiete verwendet werden.

Maven-Projekt
Dieses Profil baut ein Maven-Projekt. Jenkins wertet dabei Ihre POM Dateien aus und reduziert damit den Konfigurationsaufwand ganz erheblich.

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.

Multikonfigurationsprojekt bauen
Dieses Profil eignet sich sehr gut für Projekte mit zahlreichen Konfigurationen, die etwa in unterschiedlichen Umgebungen getestet oder plattformspezifisch gebaut werden müssen.

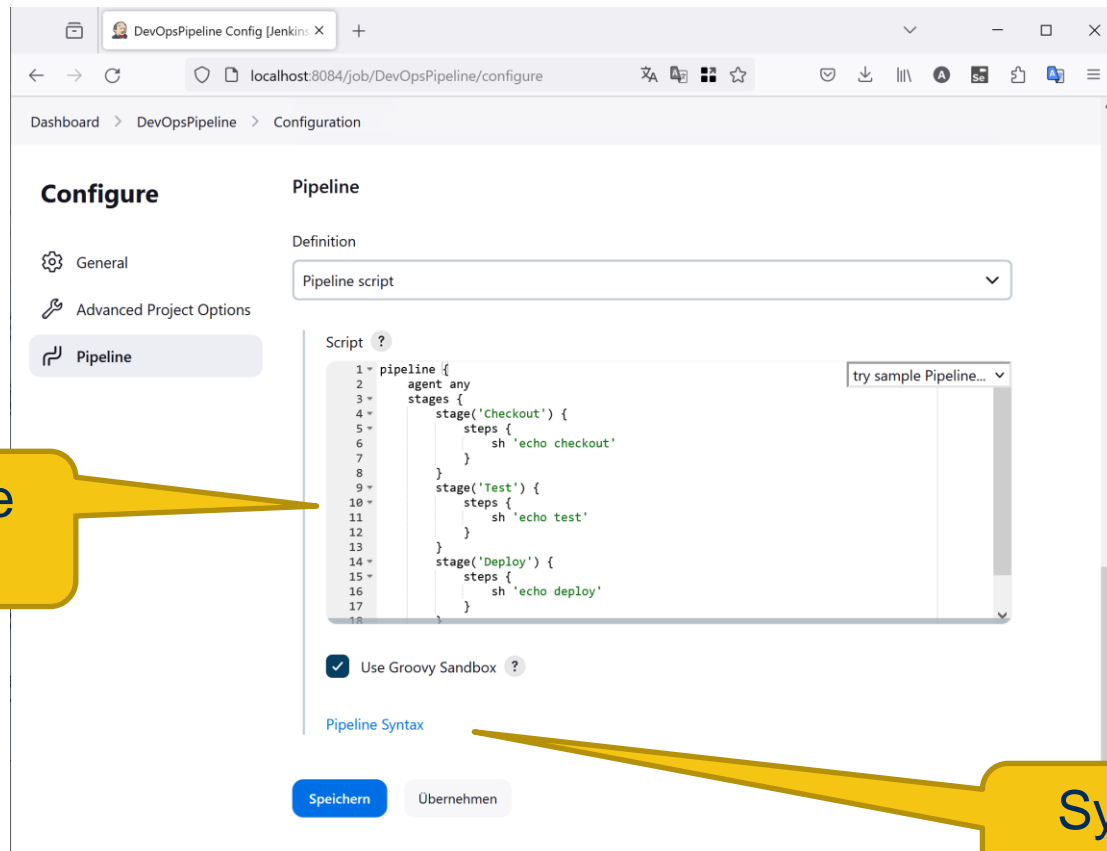
OK

Multibranch Pipeline

Pipeline
Job

Pipeline Stages & Steps Hello World

Tutorial



```
pipeline {
    agent any
    stages {
        stage('Checkout') {
            steps {
                sh 'echo checkout'
            }
        }
        stage('Test') {
            steps {
                sh 'echo test'
            }
        }
        stage('Deploy') {
            steps {
                sh 'echo deploy'
            }
        }
    }
}
```

Pipeline Script erstellen

Tutorial

Dashboard > DevOpsPipeline > Pipeline Syntax

Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL

Sample Step
checkout: Check out from version control

checkout ?

SCM
Git

Repositories ?

Repository URL ?
https://github.com/mosazhaw/DevOpsDemo

Credentials ?
- leer -
+ Add
Erweitert

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/main

Dashboard > DevOpsPipeline > Pipeline Syntax

- leer -
+ Add
Erweitert

Add Repository

Branches to build ?

Branch Specifier (blank for 'any') ?
*/main

Add Branch

Repository Browser ?
(Auto)

Additional Behaviours
Hinzufügen

☒ Include in polling? ?
☒ Include in changelog? ?

Generate Pipeline Script

Wo ist der Checkout?

The screenshot shows the Jenkins 'Pipeline Steps' view for a build named 'DevOpsPipeline #3'. The interface includes a left sidebar with navigation options like 'Console Output', 'Edit Build Information', and 'Pipeline Steps'. The main area displays a table of pipeline steps. Three yellow callout boxes highlight specific elements: 'Build' points to the build name in the breadcrumb, 'Steps' points to the 'Pipeline Steps' link in the sidebar, and 'Workspace' points to the 'Workspaces' link in the sidebar.

Step	Arguments	Status
Start of Pipeline - (3 Sekunden in block)		✓
node - (2.7 Sekunden in block)		✓
node block - (2.6 Sekunden in block)		✓
stage - (1.8 Sekunden in block)	Checkout	✓
stage block (Checkout) - (1.7 Sekunden in block)		✓
sh - (0.28 Sekunden in self)	echo checkout	✓
stage - (0.38 Sekunden in block)	Test	✓
stage block (Test) - (0.31 Sekunden in block)		✓
sh - (0.28 Sekunden in self)	echo test	✓
stage - (0.33 Sekunden in block)	Deploy	✓
stage block (Deploy) - (0.3 Sekunden in block)		✓
sh - (0.27 Sekunden in self)	echo deploy	✓

The screenshot shows the Jenkins 'Workspace' view for the same build. The breadcrumb trail indicates the path: 'Dashboard > DevOpsPipeline > #3 > Allocate node : Start > Workspace > Workspace'. The main area displays a file tree of the workspace contents, including files like '.gitignore', 'DevOps.code-workspace', 'Dockerfile', 'package-lock.json', and 'README.md'. A link at the bottom allows downloading all files as a ZIP archive.

Workspace

- Up
- Status
- Console Output
- Workspace

- .git
- .github/workflows
- backend
- frontend
- .gitignore 09.01.2024, 15:58:34 54 B
- DevOps.code-workspace 09.01.2024, 15:58:34 306 B
- Dockerfile 09.01.2024, 15:58:34 659 B
- package-lock.json 09.01.2024, 15:58:34 93 B
- README.md 09.01.2024, 15:58:34 1017 B

(Alle Dateien als ZIP-Archiv herunterladen)

Jenkins 2.426.2

Versionierung: Auch der Build ist Code - Jenkinsfile

The screenshot shows the Jenkins web interface for configuring a pipeline. The breadcrumb navigation is 'Dashboard > DevOpsPipeline > Configuration'. The left sidebar has three tabs: 'General' (selected), 'Advanced Project Options', and 'Pipeline'. The main content area is titled 'Configure' and shows the 'Pipeline' section. Under 'Definition', 'Pipeline script from SCM' is selected. Below this, there are fields for 'SCM' (set to 'Keines'), 'Script Path' (set to 'Jenkinsfile'), and a checked checkbox for 'Lightweight checkout'. At the bottom are 'Speichern' and 'Übernehmen' buttons. The footer shows 'REST API' and 'Jenkins 2.426.2'.

Script wird
aus Version
Control
geladen

Mit dieser Lösung kann
auch der Build versioniert
werden.

Build-Entwicklung analog
Code-Entwicklung.

Jenkins Clouds

Continuous Integration Advanced

Das Ausführen eines Builds kann viel Performance benötigen. Daher kann eine Cloud betrieben werden, um Builds (verschiedener Entwickler) effizient und parallel auszuführen.

Jenkins Docker Cloud

Executor

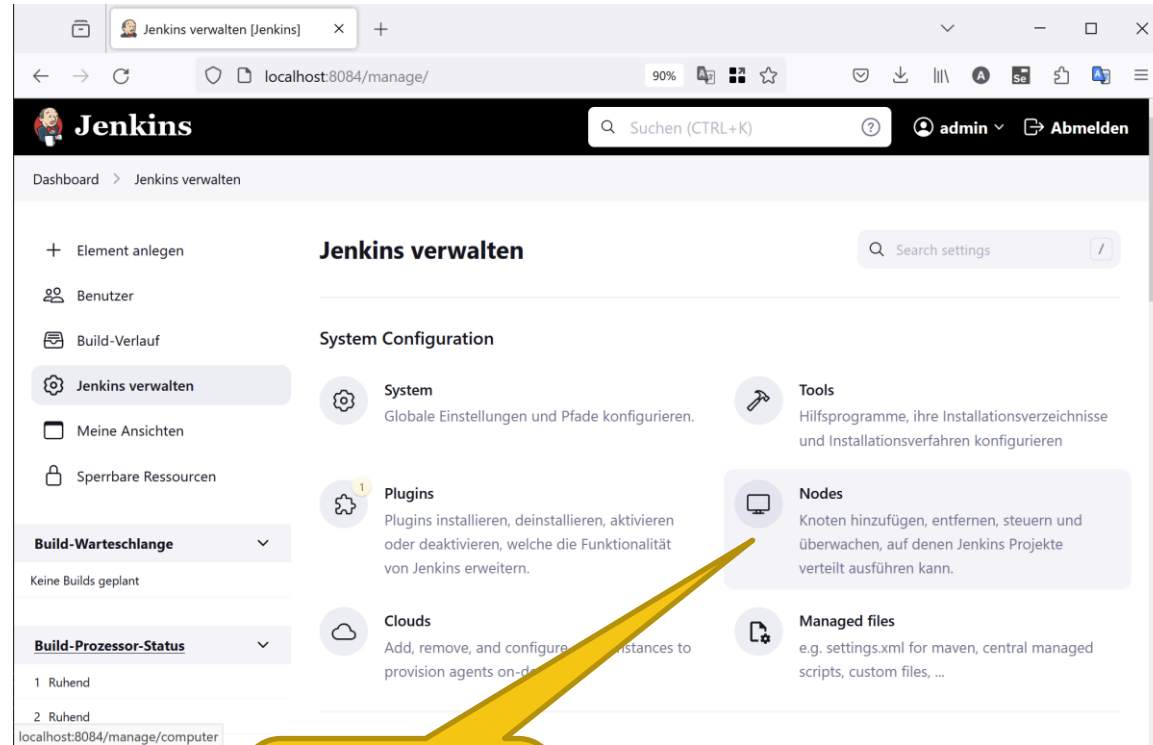
Auf einem Executor kann ein Jenkins Build ausgeführt werden. Bisher haben wir implizit den Executor «master» verwendet.

Cloud

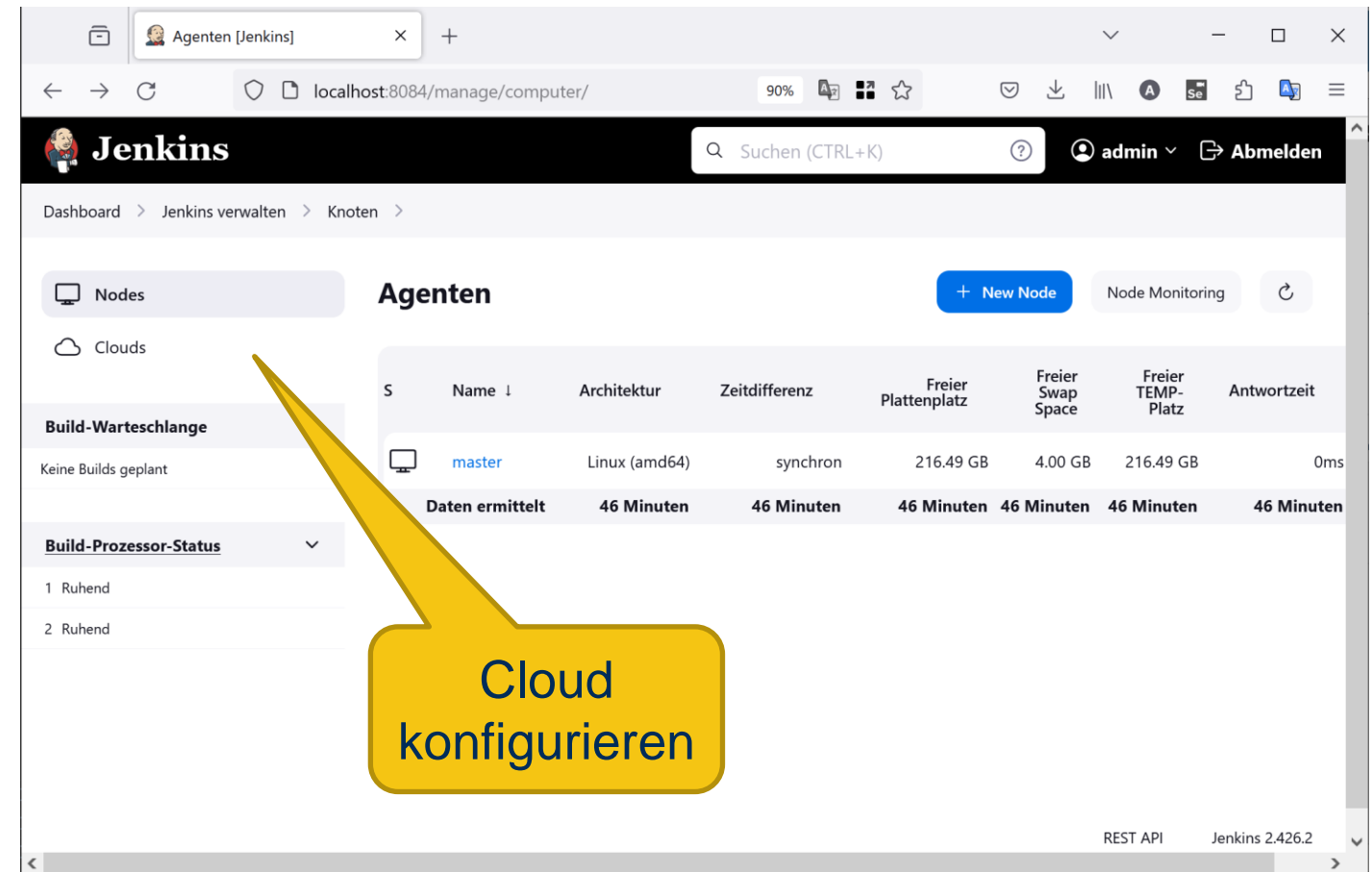
Wir fügen Jenkins eine Cloud hinzu, die auf Abruf Executoren zur Verfügung stellt. Diese Executoren sind Docker Containers, die auf einem Docker Image basieren und den Build ausführen.

Jenkins Cloud verwalten

Tutorial

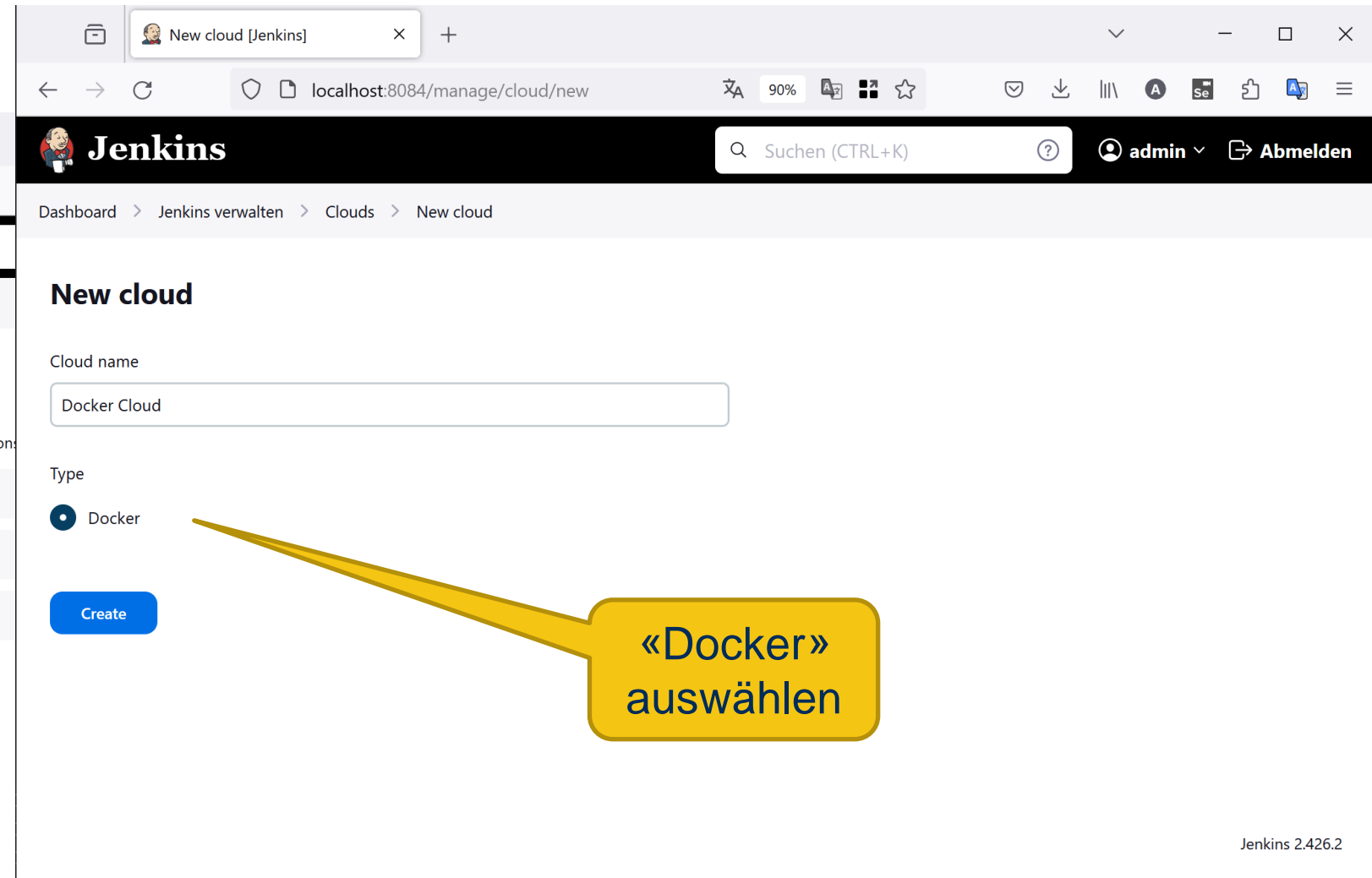
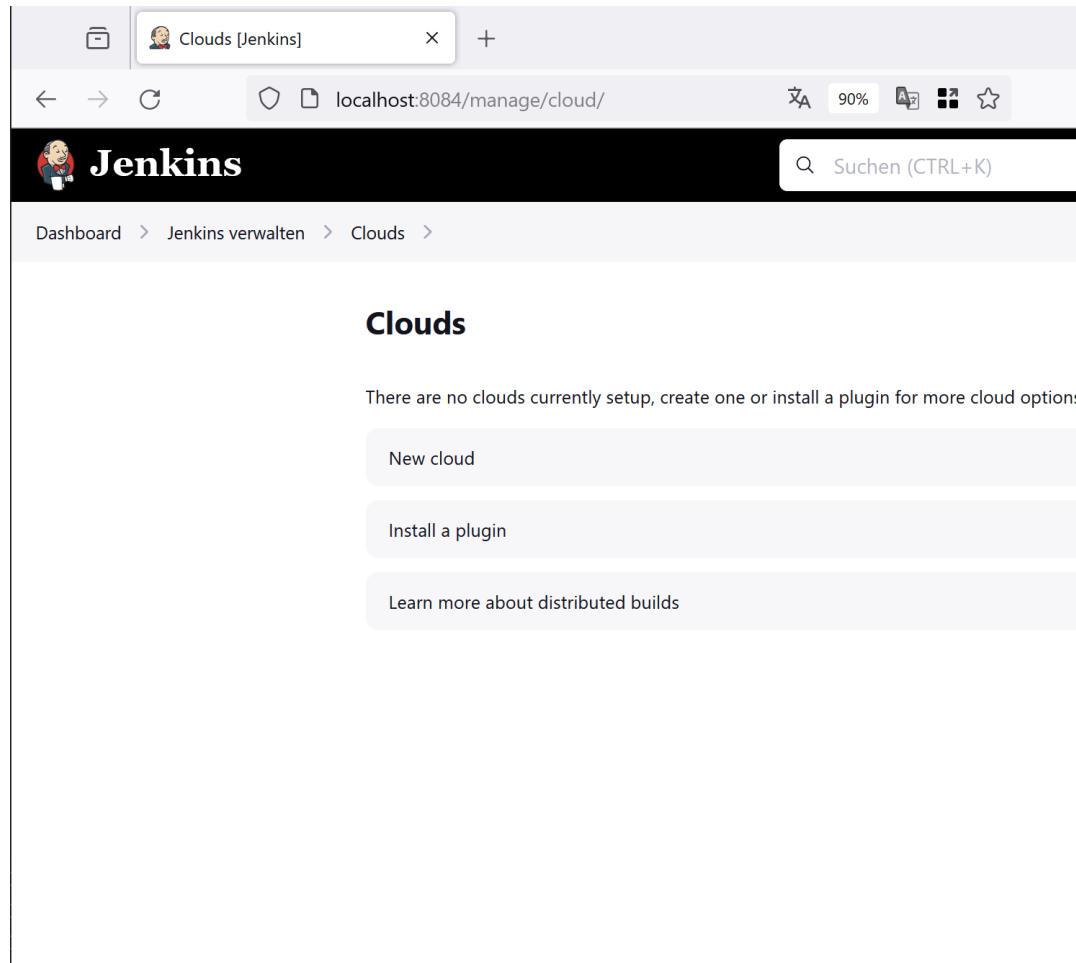


Jenkins
verwalten
→ Nodes



Cloud
konfigurieren

Eine neue Cloud zu Jenkins hinzufügen



Die Docker Cloud kann unter **tcp://host.docker.internal:2375** erreicht werden.

Mit **Test Connection** kann die Verbindung getestet werden (TLS abschalten, nächste Folie)

Verbindung zu
Docker Host
(Achtung keine
Leerzeichen am
Ende)

Aktivieren

Testen!

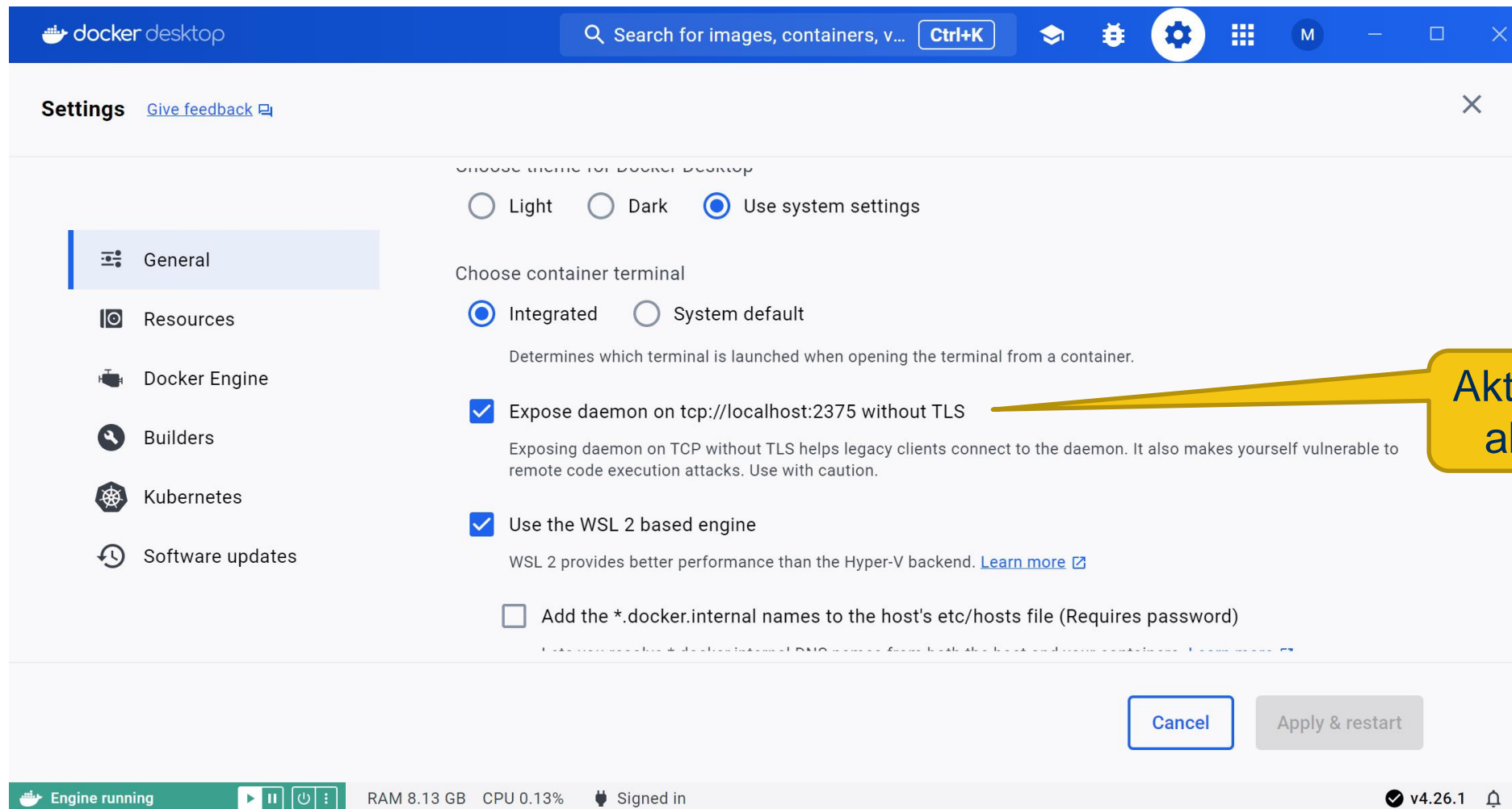
The screenshot shows the Jenkins web interface for configuring a new Docker Cloud. The browser address bar shows 'localhost:8084/manage/cloud/create'. The Jenkins header includes a search bar and user 'admin'. The breadcrumb trail is 'Dashboard > Jenkins verwalten > Clouds > New cloud'. The 'New cloud' form has the following fields:

- Name:** A text input field containing 'Docker Cloud'.
- Docker Cloud details:** A section header with an 'Edited' status.
- Docker Host URI:** A text input field containing 'tcp://host.docker.internal:2375'.
- Server credentials:** A dropdown menu showing '- leer -' with an '+ Add' button below it.
- Erweitert:** A dropdown menu.
- Version:** A label indicating 'Version = 24.0.7, API Version = 1.43'.
- Enabled:** A checkbox that is checked, with a label 'Enabled' and a help icon.
- Buttons:** A blue 'Save' button at the bottom left and a 'Test Connection' button at the bottom right.

TLS abschalten

Tutorial

TLS (Authentifizierung) soll für den Test abgeschaltet werden.



Docker Agent Template

Tutorial

Nun muss das Docker Template für den Jenkins Cloud Slave definiert werden.

Ein guter Start ist
jenkins/agent:jdk21

Achtung, wenn Image nicht vorhanden ist wird zuerst ein «pull» ausgeführt

Aktivieren

The screenshot shows the Jenkins interface for configuring a Docker Cloud. The main form is titled 'Cloud Docker Cloud Configuration'. It includes a 'Name' field with the value 'Docker Cloud'. Below this, there are two expandable sections: 'Docker Cloud details' and 'Docker Agent templates'. The 'Docker Agent templates' section is expanded, showing a list of images to be launched as agents. A modal window titled 'Docker Agent templates' is open, showing a form for adding a new template. The 'Labels' field is empty. The 'Enabled' checkbox is checked. The 'Name' field contains 'Docker Agent'. The 'Docker Image' field contains 'jenkins/agentjdk21'. At the bottom of the modal, there are 'Save' and 'Übernehmen' buttons.

Für Jenkins selbst («master») soll die Anzahl **Executors** auf 0 gesetzt werden, damit immer unsere Cloud benutzt wird.

The screenshot shows the Jenkins web interface at localhost:8084/computer/(built-in)/configure. The left sidebar contains links for Status, Konfigurieren (highlighted), Build-Verlauf, Auslastung, and Script-Konsole. The main content area shows the configuration for the master node. The 'Anzahl der Build-Prozessoren' field is set to 0. Below it are fields for 'Labels' and 'Auslastung' (set to 'Diesen Knoten so viel wie möglich verwenden'). At the bottom, there is a 'Speichern' button.

Master deaktivieren, damit Cloud verwendet wird

Build auf Cloud Agent laufen lassen

Tutorial

Vorbedingungen und Analyse

- Docker Cloud ist **enabled**, Docker Template ist **enabled**, Jenkins Master hat **0** Executors
- Problemanalyse: <http://localhost:8080/log/all>

The image displays two side-by-side screenshots. The left screenshot shows the Jenkins web interface at localhost:8084/computer/. The 'Agents' tab is active, showing a table of nodes. A yellow callout bubble points to the 'Docker Agent-0000ad6f07zf5' entry with the text 'Neuer Node'. The right screenshot shows the Docker Desktop interface. The 'Containers' tab is active, showing a list of containers. A yellow callout bubble points to the 'jenkins/agent' container with the text 'Neuer Container'. The status bar at the bottom of Docker Desktop indicates 'Engine running'.

Jenkins Agents Table:

S	Name ↓	Architektur	Zeitdifferenz	Freier Plattenplatz	Freie Swa Spac
	Docker Agent-0000ad6f07zf5	Linux (amd64)	synchron	215.42 GB	4.00
	master	Linux (amd64)	synchron	215.42 GB	4.00
	Daten ermittelt	2 Minuten 10 Sekunden	2 Minuten 10 Sekunden	2 Minuten 10 Sekunden	2 Minuten 10 Sekunden

Docker Desktop Containers Table:

Name	Image	Status	CPU (%)	Actions
jenkins-master-fs24	mosazhaw/je	Running	13.75%	
jenkins-class-fs23	innovad/jenk	Running	800%	
jenkins-master-FS23	innovad/je	Exited (143)	0%	
sweet_bhaskara	jenkins/agen	Running	0%	

macOS, Jenkins und Docker

Unter OS X kann es Probleme geben, von Jenkins auf Docker zuzugreifen. Der Workaround ist, einen weiteren Container zwischen Jenkins und Docker zu schalten der die Zugriffe weiterleitet.

```
docker pull alpine/socat
docker run -d --restart=always --name socat -p 127.0.0.1:2376:2375 -v
/var/run/docker.sock:/var/run/docker.sock alpine/socat tcp-
listen:2375,fork,reuseaddr unix-connect:/var/run/docker.sock
```

Danach IP-Adresse des Vermittlers rausfinden: `docker inspect socat`, Abschnitt "IPAddress" suchen.

Anschliessend sollte in Jenkins der Zugriff mit `tcp://X:2375` möglich sein (X = IP-Adresse des Vermittlers, z.B. `tcp://172.17.0.3:2375`)

Quellen: <https://stackoverflow.com/a/53495868>, <https://hub.docker.com/r/alpine/socat/>

Lernjournal

Ziele

- Jenkins-Build von DevOpsDemo um Frontend erweitern (NodeJS)
- Jenkins Pipelines verwenden können
- Pipelines in Git ablegen können
- Jenkins Cloud Agents verwenden können

Checkliste

- ✓ NodeJS Build aufsetzen
- ✓ Jenkins Cloud Agent aufsetzen
- ✓ DevOpsDemo-Build als Pipeline-Build umsetzen
- ✓ Pipeline-Build auf Github gespeichert
- ✓ Dokumentation der Konfiguration und der Build-Ergebnisse