



Container Extensions für z/OS (zCX)

Maïke Havemann
IBM Z Technical Specialist



z/OS Container Extensions

.. was ist zCX?

Use Cases

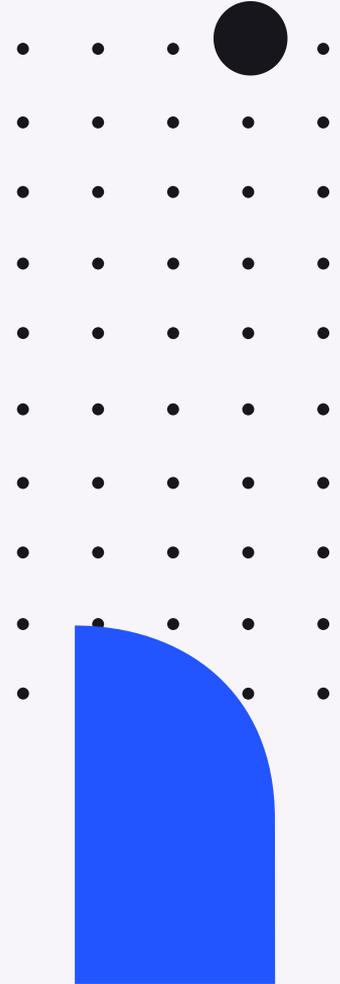
.. was theoretisch geht

DevOps

.. wenn es innovativ sein soll

Praxis

.. was in der Realität umgesetzt wird

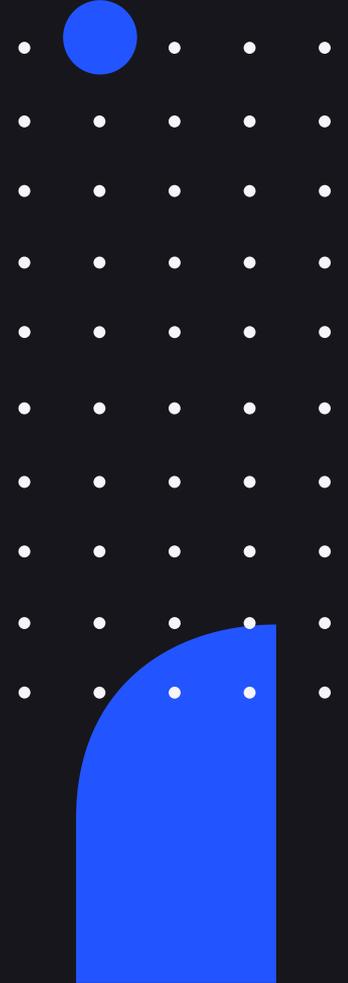


z/OS Container Extensions



ab V2.4

- **Z Linux Software als Docker Container in z/OS**
ohne separaten Linux Server
- **Maintenance über z/OS und z/OS Qualities of Service**
- **z14 oder z15 mit Container Hosting Foundation**
HW FC 0104 oder IBM Container Hosting Foundation für z/OS
- **zCX 90-Tage-Testversion**
APAR (OA58969)



z/OS Container Extensions

ab V2.4

Linux Docker Appliance

Maintenance und IBM Support, Provisionierung durch z/OSMF Workflows

Standard Docker Interfaces

Jede Software, die als Docker Image auf Linux auf Z verfügbar ist // IBM Z Registry
Kommunikation mit native z/OS Applikationen über high speed virtual IP network
Keine z/OS Skills für Entwicklung und Deployment von Docker Containern

Keine Linux Admin Skills notwendig

Interfaces auf Docker CLI limitiert, kein Zugriff auf Linux Kernel

Gemanaged als z/OS Prozess

Mehrere Instanzen in einem z/OS System
zCX Workloads sind zIIP eligible

Use Cases

Erweiterung z/OS Applikationen

- Neuste Microservices (logstash, Etc, Wordpress, etc.)
- No-SQL DBs (MongoDB, IBM Cloudant, etc.)
- Analytics Frameworks
- Messaging Frameworks (z.B. Apache Kafka)
- Web Server Proxies (z.B. nginx)
- Neuste Programmiersprachen und -umgebungen

System Management

- System Management Komponenten (bisher nicht verfügbar für z/OS)
- z.B.
 - Tivoli Enterprise Portal (TEPS)
 - Service Management Unite (SMU)

Open Source

- Ergänzen eines bestehenden z/OS, Zowe und DevOps
- Gitlab/Github server
- Linux basierte Dev Tools
- Linux Shell Environments
- ...



**Menschen >
Prozesse, Tools**



CI/CD

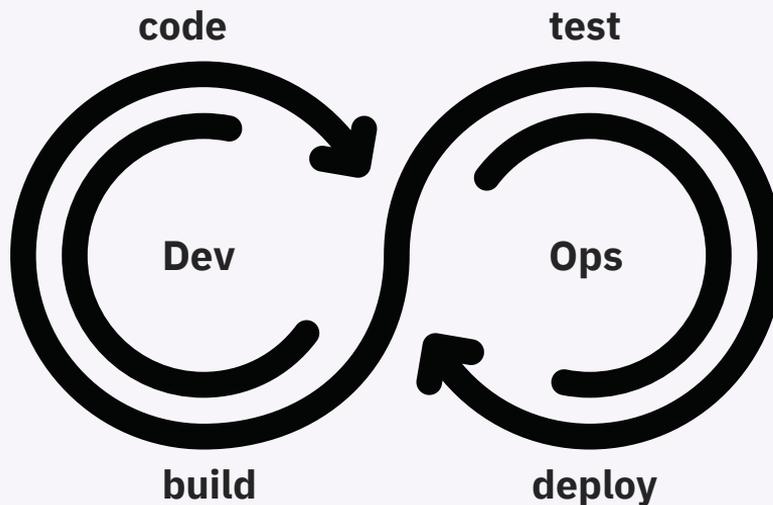


Fehlerreduzierung

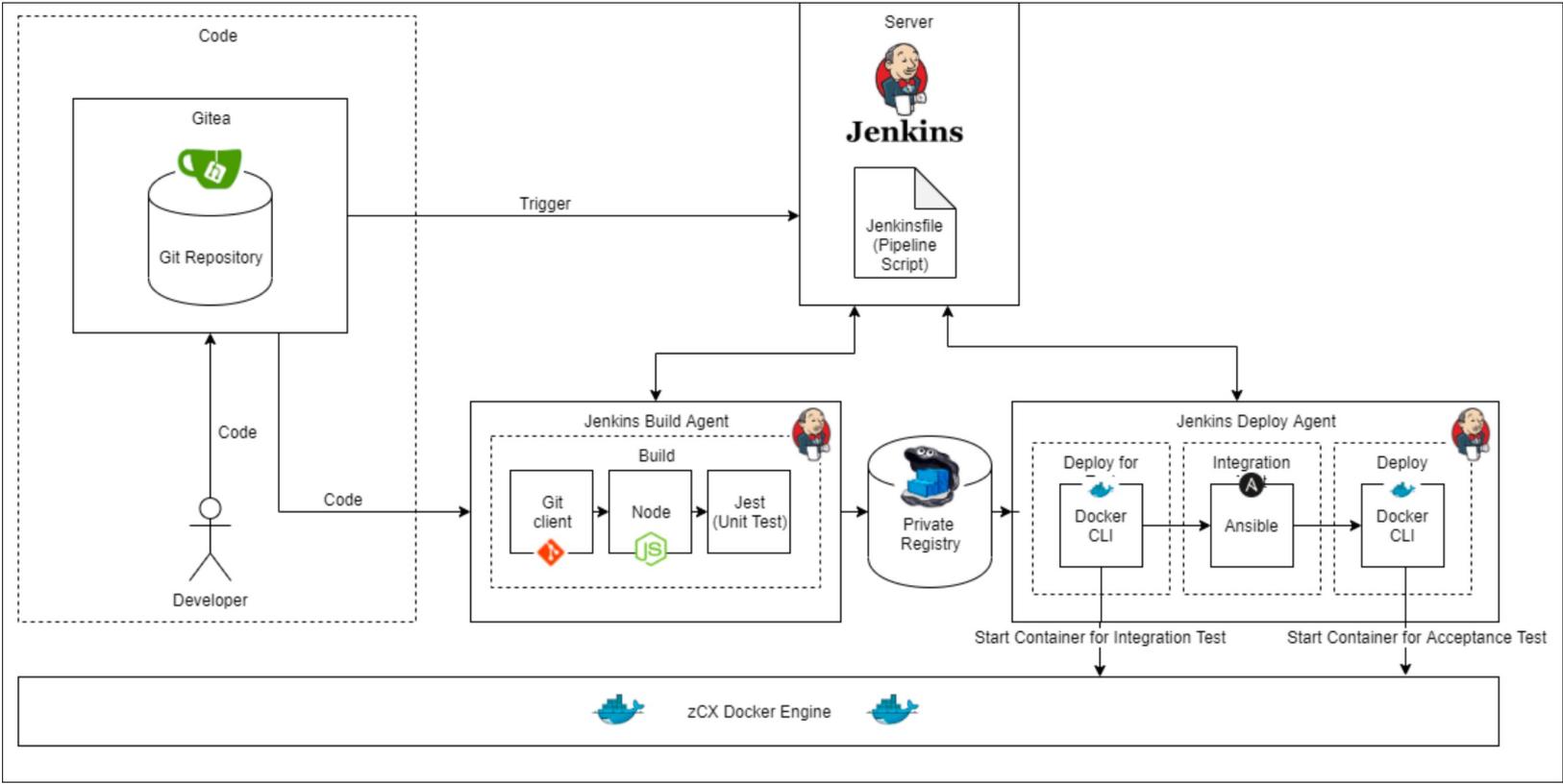


**Plattform-
unabhängig**

DevOps



DevOps Use Case / Architektur Überblick



DevOps / Gitea als Code Repository

Was ist Gitea?

- Self-hosted Git Service
- Für viele Plattformen / Architekturen
- Schnell und intuitiv aufsetzbar

Anforderungen

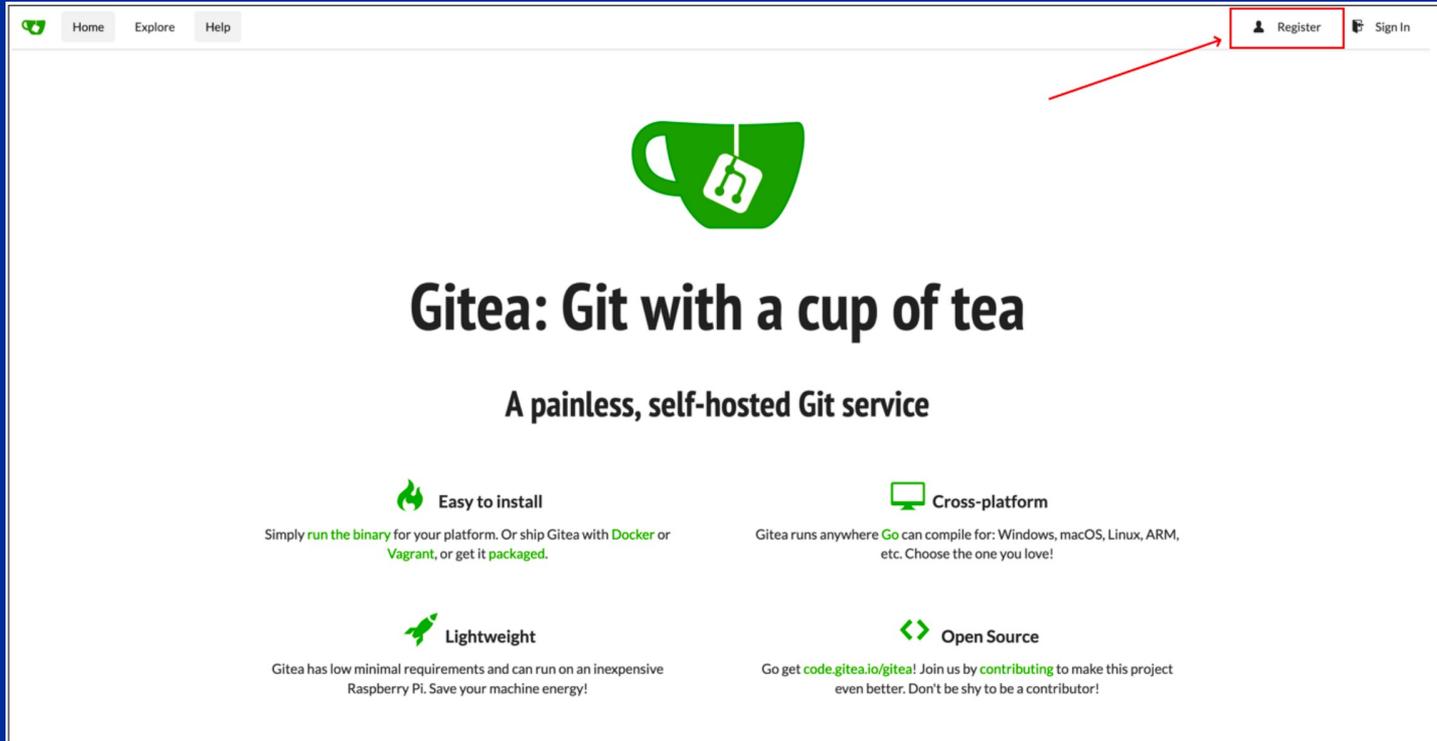
- Laufende zCX Instanz
- Volume für Gitea angelegt
- Laufender Gitea Container
- Code für eine node.js Applikation

Instruktionen: siehe Redbook



DevOps Use Case / Gitea als Code Repository

1. Gitea User Interface über Browser öffnen
2. Anmelden



Home Explore Help

Register Sign In



Gitea: Git with a cup of tea

A painless, self-hosted Git service

-  **Easy to install**
Simply [run the binary](#) for your platform. Or ship Gitea with [Docker](#) or [Vagrant](#), or get it [packaged](#).
-  **Cross-platform**
Gitea runs anywhere [Go](#) can compile for: Windows, macOS, Linux, ARM, etc. Choose the one you love!
-  **Lightweight**
Gitea has low minimal requirements and can run on an inexpensive Raspberry Pi. Save your machine energy!
-  **Open Source**
Go get [code.gitea.io/gitea!](#) Join us by [contributing](#) to make this project even better. Don't be shy to be a contributor!

DevOps Use Case / Gitea als Code Repository

3. Gitea Container einmalig konfigurieren

Initial Configuration

If you run Gitea inside Docker, please read the [documentation](#) before changing any settings.

Database Settings

Gitea requires MySQL, PostgreSQL, MSSQL or SQLite3.

Database Type *

Path *

File path for the SQLite3 database.
Enter an absolute path if you run Gitea as a service.

General Settings

Site Title *

You can enter your company name here.

Repository Root Path *

Remote Git repositories will be saved to this directory.

Git LFS Root Path

Files tracked by Git LFS will be stored in this directory. Leave empty to disable.

Run As Username *

Enter the operating system username that Gitea runs as. Note that this user must have access to the repository root path.

SSH Server Domain *

Domain or host address for SSH clone URLs.

SSH Server Port

Port number your SSH server listens on. Leave empty to disable.

Gitea HTTP Listen Port *

Port number the Gitea web server will listen on.

Gitea Base URL *

Base address for HTTP(S) clone URLs and email notifications.

Log Path *

DevOps Use Case / Gitea als Code Repository

4. Accountinformationen ausfüllen und registrieren

Register

Username *

Email Address *

Password *

Re-Type Password *

[Register Account](#)

[Already have an account? Sign in now!](#)

DevOps Use Case / Gitea als Code Repository

5. Repository für den Code einer Applikation anlegen

The screenshot displays the Gitea user interface. At the top, a green notification bar states "Account was successfully created." Below this, the "Respositories" section is active, showing "0" repositories. A red box highlights a blue "+" icon in the top right corner of the Respositories section, with a red arrow pointing to it from the notification bar. The interface also features a "Repository" and "Organization" tab, a search bar for finding repositories, and a filter menu with options: "All", "Sources", "Forks", "Mirrors", and "Collaborative".

0 total contributions in the last 12 months

	Jul	Aug	Sep	Oct	Nov	Dec	Jan	Feb	Mar	Apr	May	Jun
Mon												
Wed												
Fri												

Less More

Repository Organization

Respositories 0

Find a repository...

All Sources Forks Mirrors Collaborative

DevOps Use Case / Gitea als Code Repository

5. Repository für den Code einer Applikation anlegen

New Repository

Owner *  Maikch

Some organizations may not show up in the dropdown due to a maximum repository count limit

Repository Name * hello-node

Good repository names use short, memorable and unique keywords.

Visibility Make Repository Private

Only the owner or the organization members if they have rights, will be able to see it.

Description

Template Select a template.

Issue Labels Select an issue label set.

.gitignore Select .gitignore templates.

License Select a license file.

README Default

Initialize Repository (Adds .gitignore, License and README)

Default Branch master

Create Repository Cancel

DevOps Use Case / Gitea als Code Repository

6. Instruktionen um Code ins Repository hinzuzufügen

Creating a new repository on the command line

```
touch README.md
git init

git add README.md
git commit -m "first commit"
git remote add origin http://129.40.23.72:3008/Maikeh/node-hello.git
git push -u origin master
```

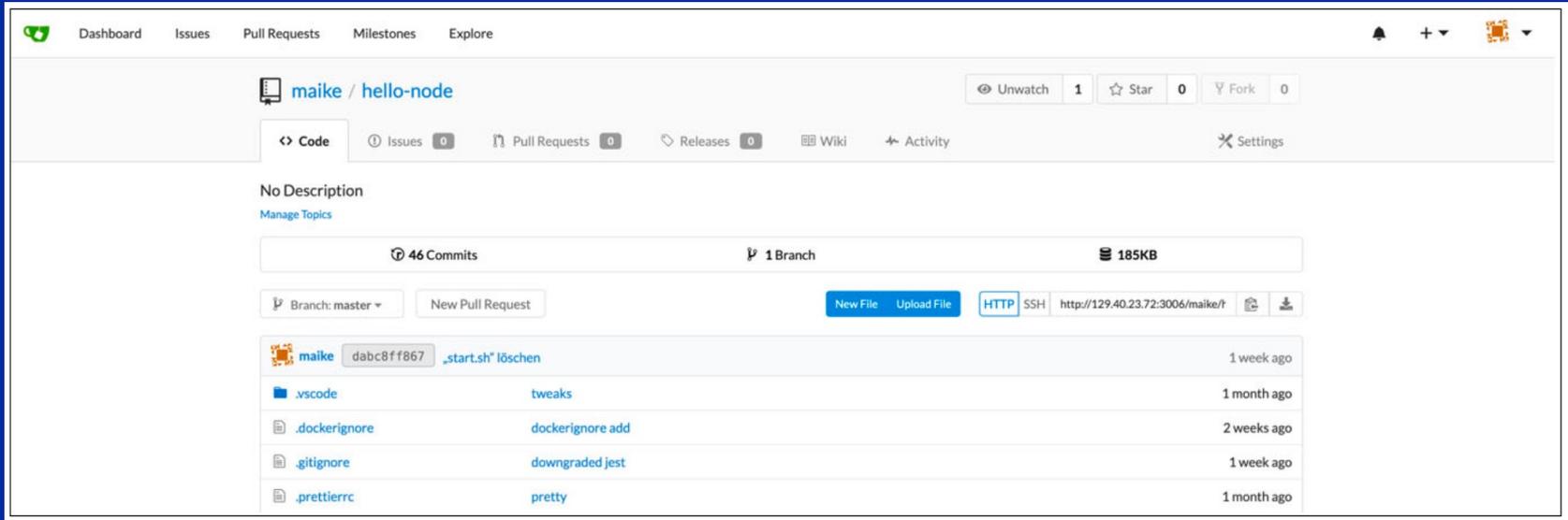
Pushing an existing repository from the command line

```
git remote add origin http://129.40.23.72:3008/Maikeh/ hello-node.git
git push -u origin master
```



DevOps Use Case / Gitea als Code Repository

7. Node.js Applikation aus dem Anhang vom Redbook in Repository pushen
Achtung: Ein Dockerfile wird benötigt



The screenshot displays the Gitea web interface for a repository named 'maike / hello-node'. The interface includes a navigation bar at the top with links for 'Dashboard', 'Issues', 'Pull Requests', 'Milestones', and 'Explore'. Below the navigation bar, the repository name 'maike / hello-node' is shown, along with statistics for 'Unwatch' (1), 'Star' (0), and 'Fork' (0). A secondary navigation bar contains links for 'Code', 'Issues' (0), 'Pull Requests' (0), 'Releases' (0), 'Wiki', 'Activity', and 'Settings'. The main content area shows 'No Description' and 'Manage Topics'. Below this, a summary bar indicates '46 Commits', '1 Branch', and '185KB'. A 'Branch: master' dropdown and a 'New Pull Request' button are visible. A 'New File' and 'Upload File' button are also present, along with a 'HTTP' button and an SSH URL 'http://129.40.23.72:3006/maike/'. A list of files is shown, including a commit by 'maike' (dabc8ff867) titled '_start.sh' löschen (1 week ago), and files like '.vscode' (tweaks, 1 month ago), '.dockerignore' (dockerignore add, 2 weeks ago), '.gitignore' (downgraded jest, 1 week ago), and '.prettierrc' (pretty, 1 month ago).

DevOps / Jenkins für automatisierte Builds

Was ist Jenkins?

- Open Source Server für eine robuste CI/CD Umgebung
- Ausführen von Jobs durch Jenkins Nodes
- Erstellen und Ausführen eines DevOps Flows durch Pipelinescript

Anforderungen

- Laufende zCX Instanz
- Volume für Jenkins angelegt
- Laufender Jenkins Container
- Admin Passwort (docker logs)

Instruktionen: siehe Redbook



DevOps Use Case / Jenkins für automatisierte Builds

1. Jenkins User Interface über Browser öffnen
2. Admin Passwort aus Docker Konsole einfügen

Getting Started

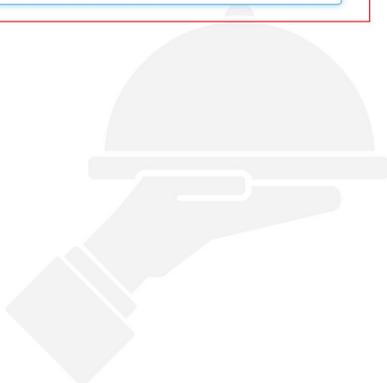
Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/root/.jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password



[Continue](#)

DevOps Use Case / Jenkins für automatisierte Builds

3. Vorgeschlagene Plugins installieren

Getting Started

Customize Jenkins

Plugins extend Jenkins with additional features to support many different needs.

Install suggested plugins

Install plugins the Jenkins community finds most useful.

Select plugins to install

Select and install plugins most suitable for your needs.

Jenkins 2.235.1

DevOps Use Case / Jenkins für automatisierte Builds

4. Admin User erstellen

Getting Started

Create First Admin User

Username:

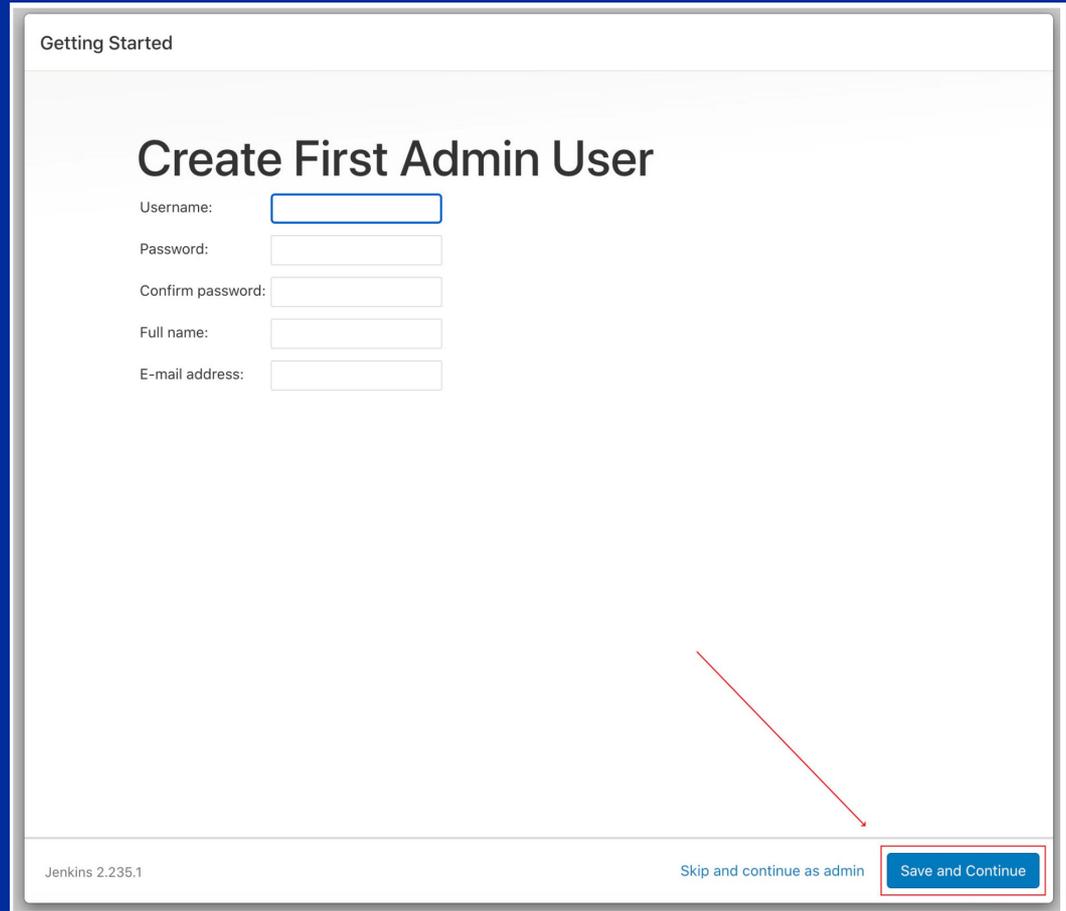
Password:

Confirm password:

Full name:

E-mail address:

Jenkins 2.235.1 [Skip and continue as admin](#)



DevOps Use Case / Jenkins für automatisierte Builds

5. Jenkins URL in die Instanz Konfiguration einfügen

Getting Started

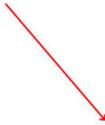
Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `BUILD_URL` environment variable provided to build steps.

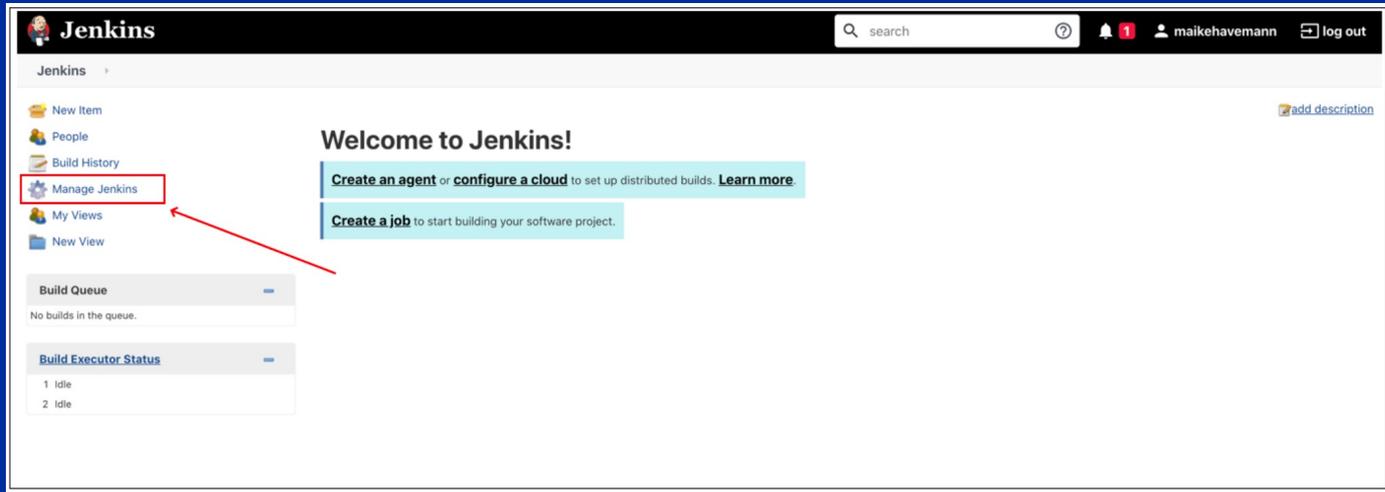
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.235.1 Not now



DevOps Use Case / Jenkins für automatisierte Builds

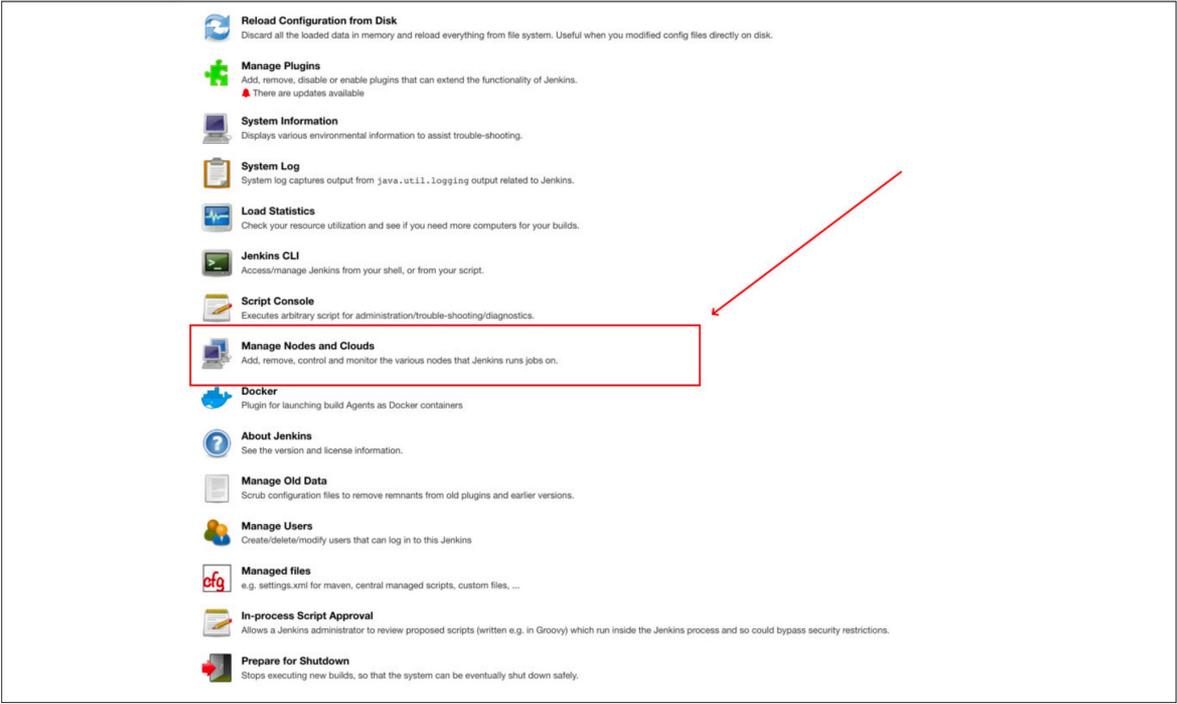
6. Ins Manage Jenkins Menü gehen



The screenshot displays the Jenkins web interface. At the top, there is a search bar and user information for 'maikehavemann'. The main navigation menu on the left includes 'New Item', 'People', 'Build History', 'Manage Jenkins' (highlighted with a red box and a red arrow), 'My Views', and 'New View'. The central area features a 'Welcome to Jenkins!' message with two primary actions: 'Create an agent or configure a cloud to set up distributed builds. Learn more.' and 'Create a job to start building your software project.'. Below the welcome message, there are two status panels: 'Build Queue' showing 'No builds in the queue.' and 'Build Executor Status' showing two 'Idle' executors.

DevOps Use Case / Jenkins für automatisierte Builds

7. Auf *Manage Nodes and Clouds* klicken



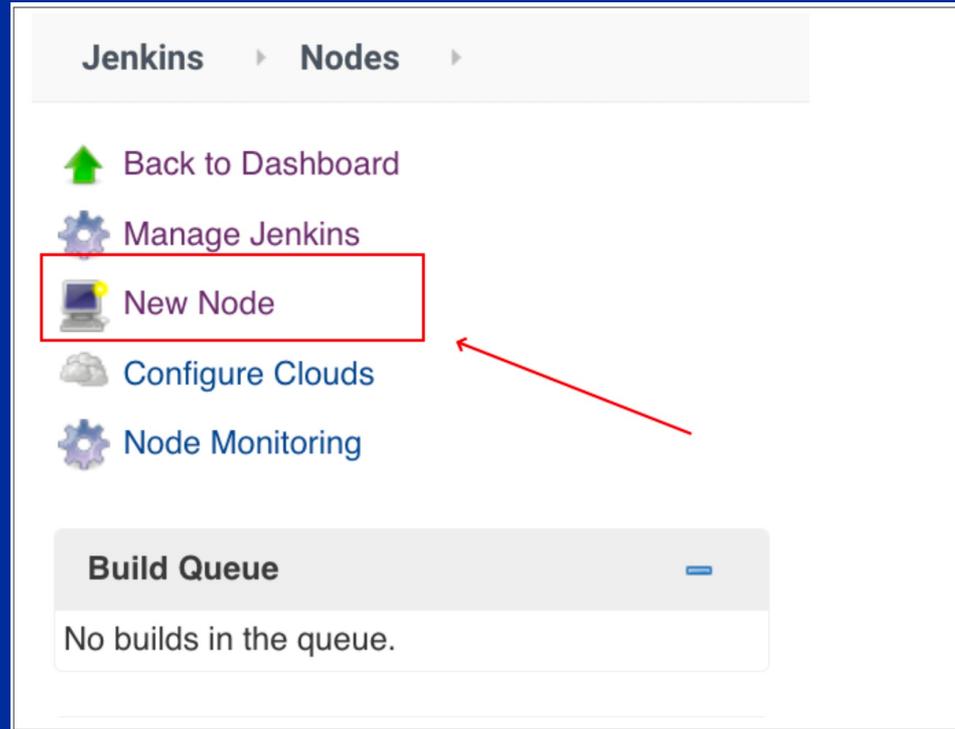
The screenshot shows the Jenkins administration interface with the following items listed:

- 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.
▲ There are updates available
- 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.
- Jenkins CLI**
Access/manage Jenkins from your shell, or from your script.
- Script Console**
Executes arbitrary script for administration/trouble-shooting/diagnostics.
- Manage Nodes and Clouds**
Add, remove, control and monitor the various nodes that Jenkins runs jobs on.
- Docker**
Plugin for launching build Agents as Docker containers.
- About Jenkins**
See the version and license information.
- Manage Old Data**
Scrub configuration files to remove remnants from old plugins and earlier versions.
- Manage Users**
Create/delete/modify users that can log in to this Jenkins
- Managed files**
e.g. `settings.xml` for maven, central managed scripts, custom files, ...
- In-process Script Approval**
Allows a Jenkins administrator to review proposed scripts (written e.g. in Groovy) which run inside the Jenkins process and so could bypass security restrictions.
- Prepare for Shutdown**
Stops executing new builds, so that the system can be eventually shut down safely.

A red box highlights the **Manage Nodes and Clouds** item, and a red arrow points to it from the right side of the image.

DevOps Use Case / Jenkins für automatisierte Builds

8. Neuen Node anlegen



DevOps Use Case / Jenkins für automatisierte Builds

9. Build Node konfigurieren

The screenshot shows the Jenkins configuration page for a Build Node. The form is titled "Name" and contains the following fields and options:

- Name:** j-build-agent
- Description:** (empty)
- # of executors:** 1
- Remote root directory:** /home/jenkins/agent/
- Labels:** build
- Usage:** Use this node as much as possible
- Launch method:** Launch agent by connecting it to the master
- Disable WorkDir
- Custom WorkDir path:** (empty)
- Internal data directory:** remoting
- Fail if workspace is missing
- Use WebSocket
- Availability:** Keep this agent online as much as possible

There is an "Advanced..." button on the right side of the form. Below the main configuration, there is a section titled "Node Properties" with the following options:

- Environment variables
- Tool Locations
- Disable deferred wipeout on this node

A "Save" button is located at the bottom left of the form.

DevOps Use Case / Jenkins für automatisierte Builds

10. Node Secret kopieren um den Docker Container für den Jenkins Node zu starten

 **Agent j-build-agent** Mark this node temporarily offline

Connect agent to Jenkins one of these ways:

-  Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://129.40.23.72:3000/computer/j-build-agent/slave-agent.jnlp -secret a78b9b1a263c3ada54768740619279df4f205625613a839b1ffdbe6501105b96 -workDir "/home/jenkins/agent/"
```

Run from agent command line, with the secret stored in a file:

```
echo a78b9b1a263c3ada54768740619279df4f205625613a839b1ffdbe6501105b96 > secret-file  
java -jar agent.jar -jnlpUrl http://129.40.23.72:3000/computer/j-build-agent/slave-agent.jnlp -secret @secret-file -workDir "/home/jenkins/agent/"
```

Labels

[build](#)

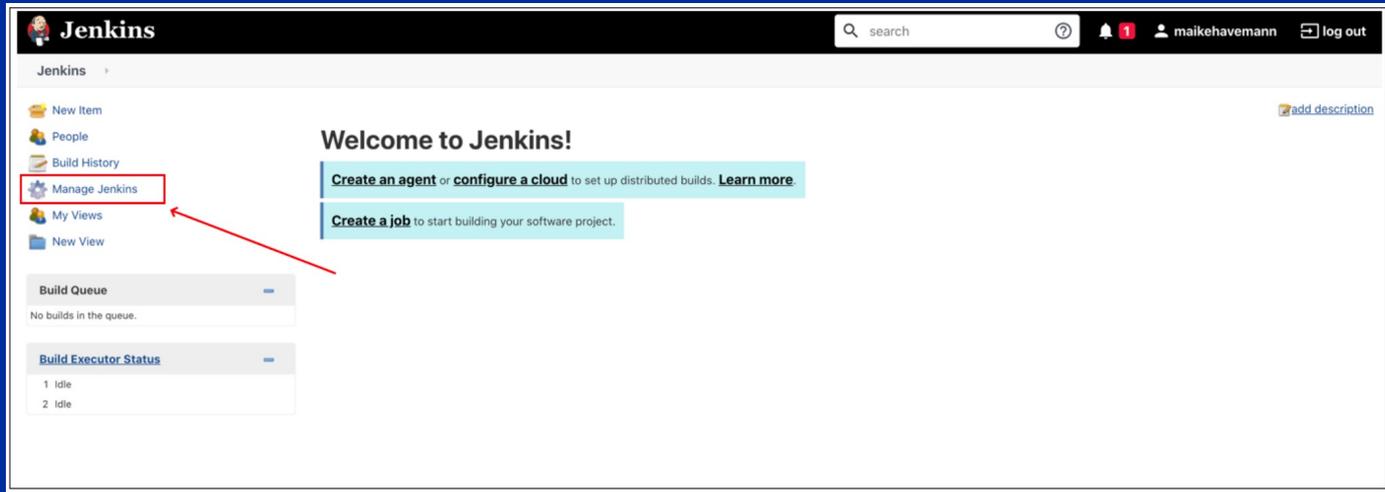
Projects tied to j-build-agent

None

DevOps Use Case / Jenkins für automatisierte Builds

Kommunikation zwischen Jenkins Server und Build Node sicherstellen

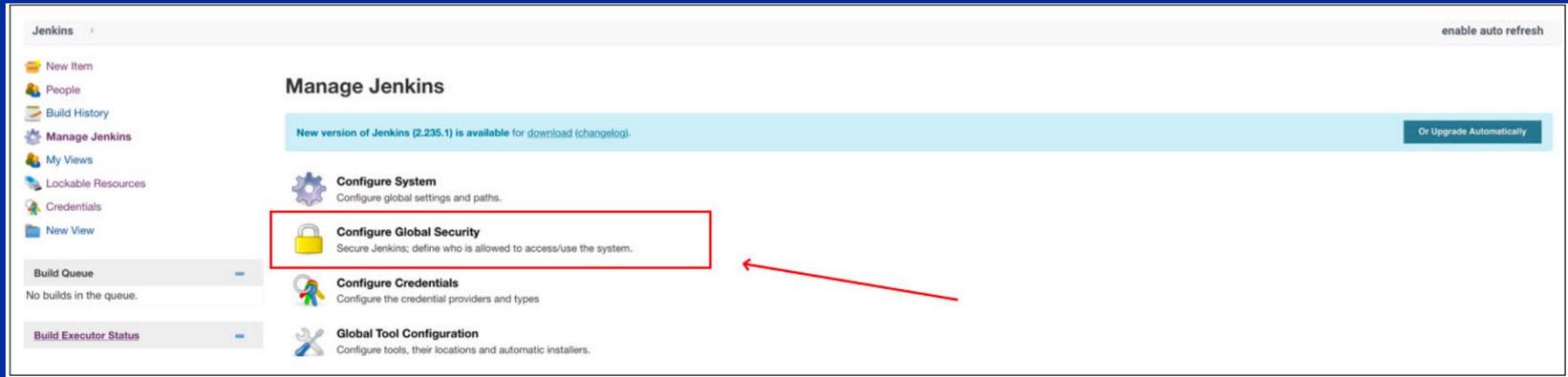
11. Ins Manage Jenkins Menü gehen



The screenshot shows the Jenkins web interface. The top navigation bar includes the Jenkins logo, a search bar, a help icon, a notification bell with a red '1', the user name 'maikehavemann', and a 'log out' button. The main content area features a 'Welcome to Jenkins!' message with two primary actions: 'Create an agent or configure a cloud to set up distributed builds. Learn more.' and 'Create a job to start building your software project.' On the left sidebar, the 'Manage Jenkins' menu item is highlighted with a red box, and a red arrow points from this box to the 'Create an agent or configure a cloud' button in the main content area. Below the sidebar, there are sections for 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing two 'Idle' executors).

DevOps Use Case / Jenkins für automatisierte Builds

12. Sicherheitseinstellungen konfigurieren



The screenshot shows the Jenkins web interface. On the left is a sidebar with navigation options: 'New Item', 'People', 'Build History', 'Manage Jenkins', 'My Views', 'Lockable Resources', 'Credentials', 'New View', 'Build Queue', and 'Build Executor Status'. The main content area is titled 'Manage Jenkins' and features a notification banner for a new Jenkins version (2.235.1). Below the banner are four configuration options: 'Configure System', 'Configure Global Security', 'Configure Credentials', and 'Global Tool Configuration'. The 'Configure Global Security' option is highlighted with a red rectangular box, and a red arrow points to it from the right.

Jenkins enable auto refresh

Manage Jenkins

New version of Jenkins (2.235.1) is available for [download](#) ([changelog](#)). Or Upgrade Automatically

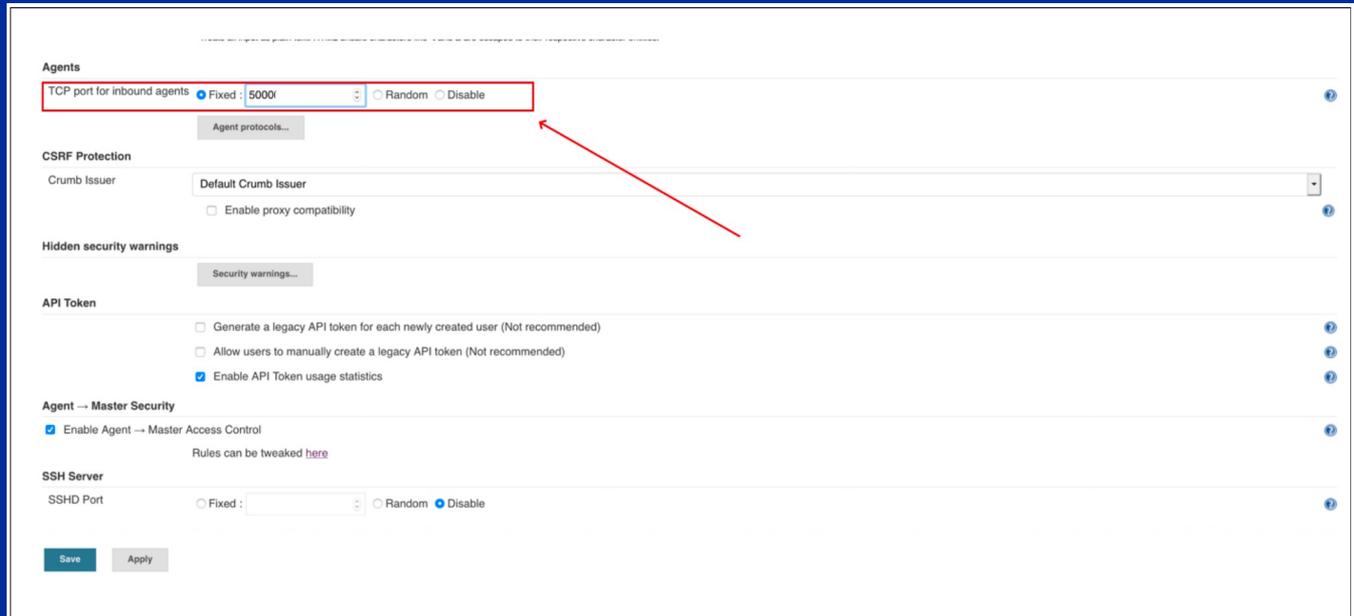
-  **Configure System**
Configure global settings and paths.
-  **Configure Global Security**
Secure Jenkins; define who is allowed to access/use the system.
-  **Configure Credentials**
Configure the credential providers and types.
-  **Global Tool Configuration**
Configure tools, their locations and automatic installers.

DevOps Use Case / Jenkins für automatisierte Builds

13. TCP Port auf 5000 (private registry) setzen und den Port beim Start vom Node Container mitgeben

14. Jenkins Build Node Container starten

Instruktionen: siehe Redbook



The screenshot shows the Jenkins configuration page for agents. The 'Agents' section is highlighted with a red box, and a red arrow points to the 'Fixed : 5000' option. The configuration includes the following sections:

- Agents**
 - TCP port for inbound agents: Fixed : 5000 Random Disable
 - Agent protocols...
- CSRF Protection**
 - Crumb Issuer: Default Crumb Issuer
 - Enable proxy compatibility
- Hidden security warnings**
 - Security warnings...
- API Token**
 - Generate a legacy API token for each newly created user (Not recommended)
 - Allow users to manually create a legacy API token (Not recommended)
 - Enable API Token usage statistics
- Agent → Master Security**
 - Enable Agent → Master Access Control
 - Rules can be tweaked [here](#)
- SSH Server**
 - SSHD Port: Fixed : Random Disable

Buttons: Save, Apply

DevOps / Ansible für automatisiertes Deployment und Tests

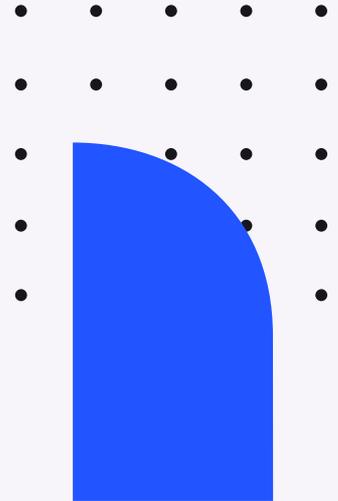
Was ist Ansible

- Automations Engine für z.B. Deployment von Applikationen
- Einfaches Deployment durch YAML Playbooks
- Kann standalone oder als Plugin in Jenkins verwendet werden

Anforderungen

- Laufende zCX Instanz
- Laufender Jenkins Container
- Ansible Playbooks für Tests und Deployment (abgelegt im Gitea Repository)

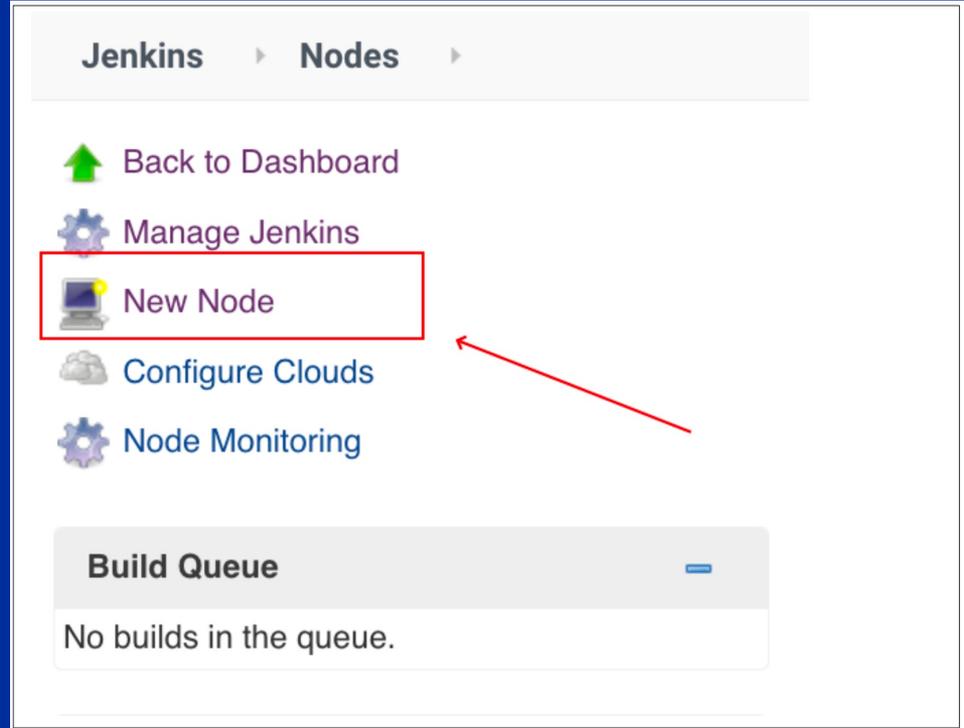
Instruktionen: siehe Redbook



DevOps Use Case / Ansible für automatisiertes Deployment und Tests

Ansible in Jenkins als Deploy Node aufsetzen

1. Ins Manage Jenkins Menü gehen
2. Auf Manage Nodes and Clouds klicken
3. Neuen Node anlegen



DevOps Use Case / Ansible für automatisiertes Deployment und Tests

Ansible in Jenkins als Deploy Node aufsetzen

4. Jenkins Deploy Node konfigurieren

The screenshot shows the Jenkins configuration page for a new Deploy Node. The form is titled "Name" and contains the following fields and options:

- Name:** j-deploy-agent
- Description:** (empty)
- # of executors:** 1
- Remote root directory:** /home/jenkins/agent/
- Labels:** test
- Usage:** Use this node as much as possible
- Launch method:** Launch agent by connecting it to the master
 - Disable WorkDir
- Custom WorkDir path:** (empty)
- Internal data directory:** remoting
 - Fail if workspace is missing
 - Use WebSocket
- Availability:** Keep this agent online as much as possible

At the bottom of the form, there is a section for "Node Properties" with the following options:

- Environment variables
- Tool Locations
- Disable deferred wipeout on this node

A "Save" button is located at the bottom left of the form.

DevOps Use Case / Ansible für automatisiertes Deployment und Tests

Ansible in Jenkins als Deploy Node aufsetzen

5. Node Secret kopieren und den Docker Container für den Deploy Node zu starten

Instruktionen: siehe Redbook



Agent j-deploy-agent

Mark this node temporarily offline

Connect agent to Jenkins one of these ways:

-  Launch agent from browser
- Run from agent command line:

```
java -jar agent.jar -jnlpUrl http://129.40.23.72:3000/computer/j-deploy-agent/slave-agent.jnlp -secret 34747350a14f981402816f91c731f7067860aa33923f133e7064722c6c2e7e64 -workDir "/home/jenkins/agent/"
```

Run from agent command line, with the secret stored in a file:

```
echo 34747350a14f981402816f91c731f7067860aa33923f133e7064722c6c2e7e64 > secret-file  
java -jar agent.jar -jnlpUrl http://129.40.23.72:3000/computer/j-deploy-agent/slave-agent.jnlp -secret @secret-file -workDir "/home/jenkins/agent/"
```

Labels

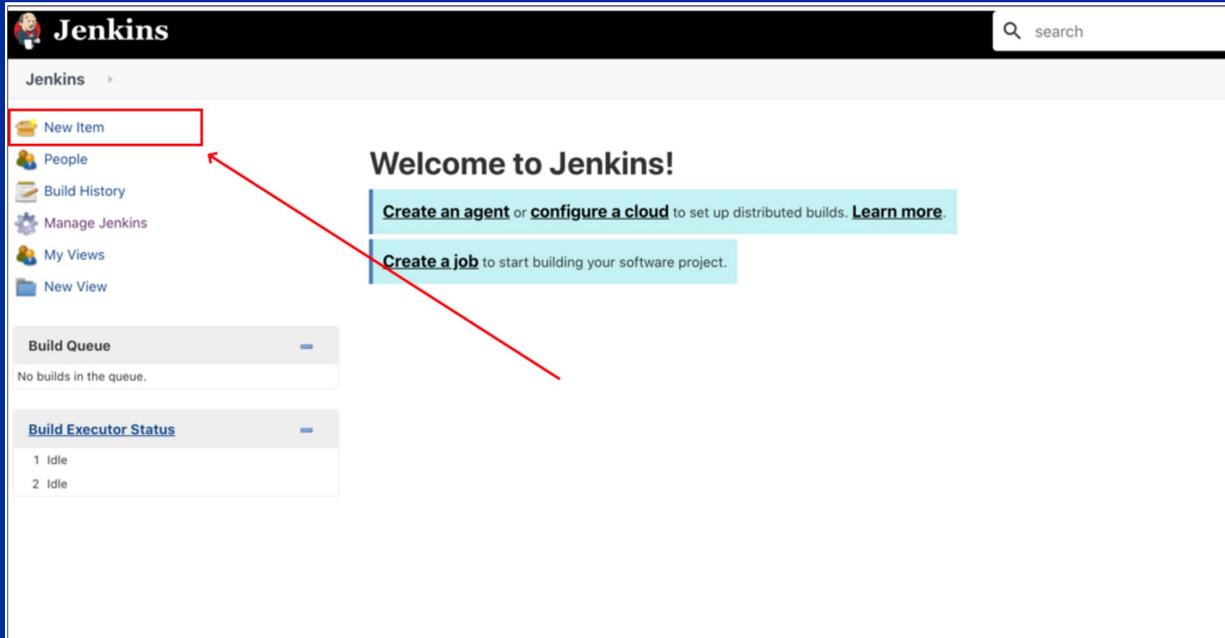
[test](#)

Projects tied to j-deploy-agent

None

DevOps Use Case / Erstellen einer Pipeline

1. Neues Item im Jenkins Interface anlegen



The screenshot displays the Jenkins web interface. The top navigation bar includes the Jenkins logo and a search field. The left sidebar contains a menu with the following items: 'New Item' (highlighted with a red box), 'People', 'Build History', 'Manage Jenkins', 'My Views', and 'New View'. The main content area features a 'Welcome to Jenkins!' message with two instructional boxes: 'Create an agent or configure a cloud to set up distributed builds. [Learn more.](#)' and 'Create a job to start building your software project.' Below the welcome message, there are two status sections: 'Build Queue' (showing 'No builds in the queue.') and 'Build Executor Status' (showing '1 Idle' and '2 Idle'). A red arrow points from the 'New Item' button to the 'Create a job' instruction box.

DevOps Use Case / Erstellen einer Pipeline

2. Item benennen und Pipeline auswählen

Enter an item name

* 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

 **Pipeline**
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or 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.

 **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 same name as long as they are in different folders.

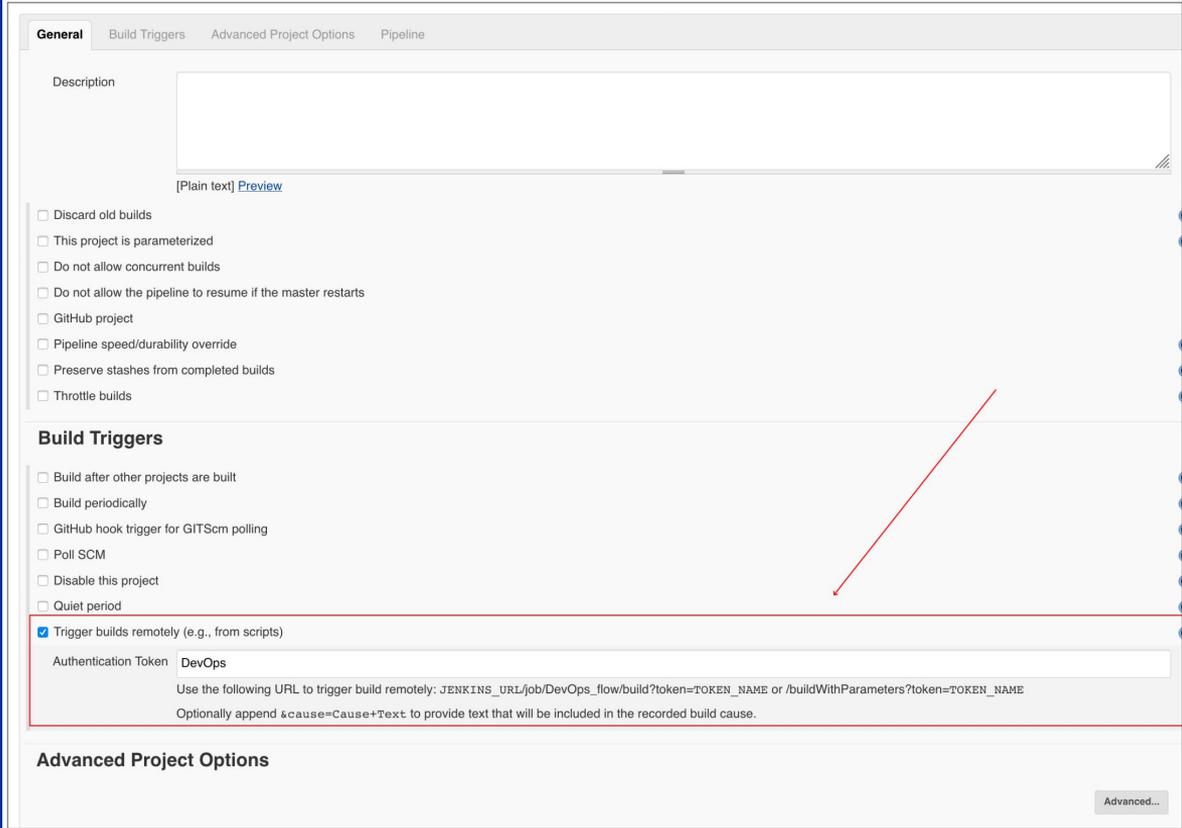
 **GitHub Organization**
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

 **Gitea Organization**
Scans a Gitea Organization (or user account) for all repositories matching some defined markers.

 **Multibranch Pipeline**

DevOps Use Case / Erstellen einer Pipeline

3. Aktivieren eines Triggers um mit jedem Code push auf die Applikation die Pipeline auszulösen



The screenshot shows the Jenkins configuration interface for a pipeline. The 'Build Triggers' section is highlighted with a red border and a red arrow pointing to the 'Trigger builds remotely' checkbox, which is checked. Below this, the 'Authentication Token' is set to 'DevOps'. The 'Advanced Project Options' section is visible at the bottom.

General Build Triggers Advanced Project Options Pipeline

Description

[Plain text] [Preview](#)

- Discard old builds
- This project is parameterized
- Do not allow concurrent builds
- Do not allow the pipeline to resume if the master restarts
- GitHub project
- Pipeline speed/durability override
- Preserve stashes from completed builds
- Throttle builds

Build Triggers

- Build after other projects are built
- Build periodically
- GitHub hook trigger for GITScm polling
- Poll SCM
- Disable this project
- Quiet period
- Trigger builds remotely (e.g., from scripts)

Authentication Token

Use the following URL to trigger build remotely: `JENKINS_URL/job/DevOps_flow/build?token=TOKEN_NAME` or `/buildWithParameters?token=TOKEN_NAME`
Optionally append `&cause=Cause+Text` to provide text that will be included in the recorded build cause.

Advanced Project Options

[Advanced...](#)

DevOps Use Case / Erstellen einer Pipeline

4. Steuerung der Pipeline durch Skript (“Jenkinsfile”), abgelegt in Gitea Repository

The screenshot shows the Jenkins Pipeline configuration page. The following fields are highlighted with red boxes and red arrows:

- Definition:** Pipeline script from SCM
- SCM:** Git
- Repositories:** Repository URL: `http://129.40.23.72:3008/maike/hello-node.git`
- Branches to build:** Branch Specifier (blank for 'any'): `*/master`
- Script Path:** Jenkinsfile

Other visible fields include:

- Credentials: - none -
- Repository browser: (Auto)
- Additional Behaviours: Add
- Lightweight checkout:

DevOps Use Case / Erstellen einer Pipeline

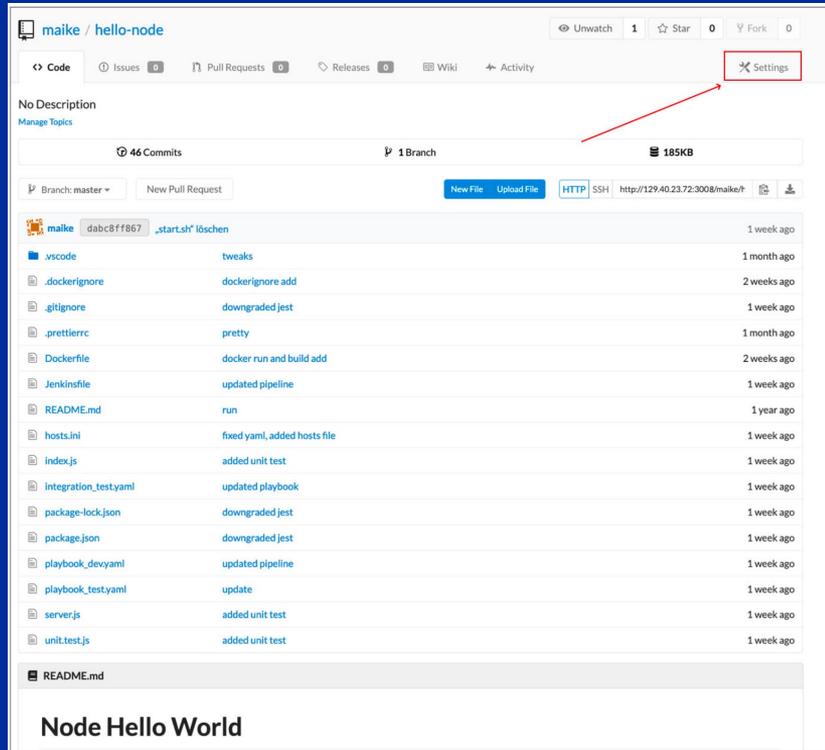
Jenkinsfile

```
pipeline {
    agent none
    stages {
        stage('Build') {
            agent{
                node {
                    label 'build'
                }
            }
            steps {
                checkout scm
                sh 'npm install'
                sh 'npm test'
                sh 'docker build -t localhost:5000/hello-node:latest .'
                sh 'docker push localhost:5000/hello-node'
            }
        }
        stage('Deploy') {
            agent{
                node {
                    label 'test'
                }
            }
            steps{
                ansiblePlaybook(inventory: 'hosts.ini', playbook: 'playbook_dev.yaml')
                ansiblePlaybook(inventory: 'hosts.ini', playbook: 'integration_test.yaml')
                ansiblePlaybook(inventory: 'hosts.ini', playbook: 'playbook_test.yaml')
            }
        }
    }
}
```

DevOps Use Case / Erstellen einer Pipeline

Webhook in Gitea erstellen

1. Im Gitea Repository, ins Einstellungen Menü



The screenshot shows the Gitea repository interface for 'maike / hello-node'. The 'Settings' menu item is highlighted with a red box and a red arrow pointing to it. The repository shows 46 commits, 1 branch, and 185KB of files. A list of files and their commit hashes is visible, including .vscode, .dockerignore, .gitignore, .prettierrc, Dockerfile, Jenkinsfile, README.md, hosts.ini, index.js, integration_test.yaml, package-lock.json, package.json, playbook_dev.yaml, playbook_test.yaml, server.js, and unit.test.js. The README.md content is 'Node Hello World'.

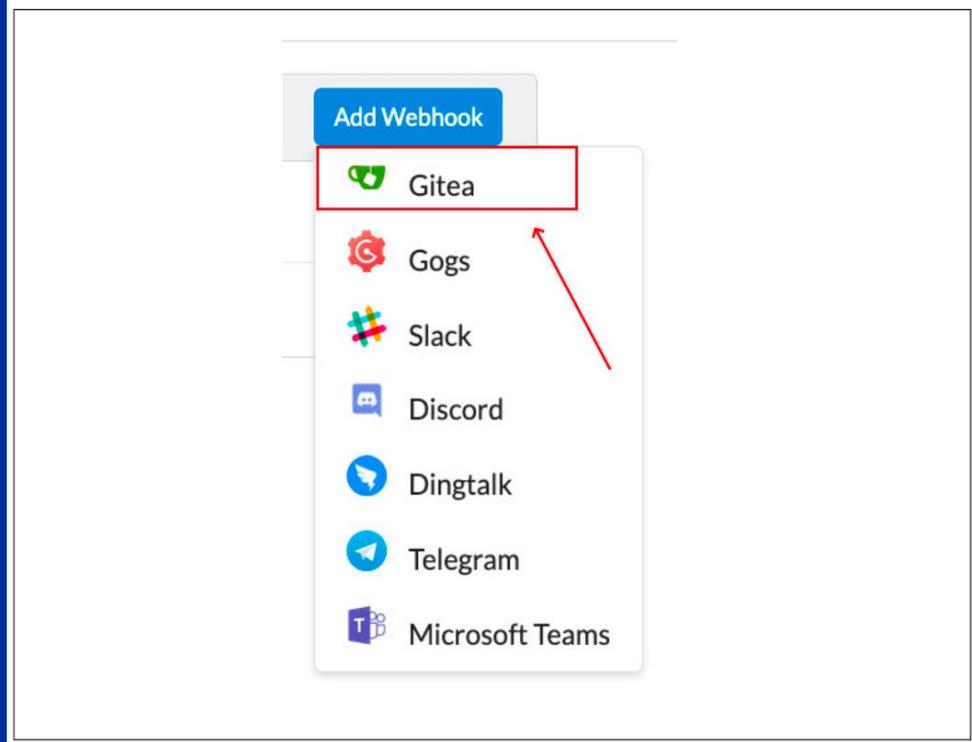
File	Commit Hash	Commit Message	Time Ago
malike	dabc8ff867	„start.sh“ löschen	1 week ago
.vscode		tweaks	1 month ago
.dockerignore		dockerignore add	2 weeks ago
.gitignore		downgraded jest	1 week ago
.prettierrc		pretty	1 month ago
Dockerfile		docker run and build add	2 weeks ago
Jenkinsfile		updated pipeline	1 week ago
README.md		run	1 year ago
hosts.ini		fixed yaml, added hosts file	1 week ago
index.js		added unit test	1 week ago
integration_test.yaml		updated playbook	1 week ago
package-lock.json		downgraded jest	1 week ago
package.json		downgraded jest	1 week ago
playbook_dev.yaml		updated pipeline	1 week ago
playbook_test.yaml		update	1 week ago
server.js		added unit test	1 week ago
unit.test.js		added unit test	1 week ago

Node Hello World

DevOps Use Case / Erstellen einer Pipeline

Webhook in Gitea erstellen

2. Anlegen eines Gitea Webhooks



DevOps Use Case / Erstellen einer Pipeline

Webhook in Gitea erstellen

3. Webhook mit Jenkins Pipeline Trigger konfigurieren

Add Webhook

Gitea will send POST requests with a specified content type to the target URL. Read more in the [webhooks guide](#).

Target URL *
https://129.40.23.72:3000/job/DevOps_flow/build?token=DevOps

HTTP Method
POST

POST Content Type
application/json

Secret

Trigger On:

Push Events
 All Events
 Custom Events...

Branch filter
*

Branch whitelist for push, branch creation and branch deletion events, specified as glob pattern. If empty or *, events for all branches are reported. See github.com/gobwas/glob documentation for syntax. Examples: master, {master, release}.

Active
Information about triggered events will be sent to this webhook URL.

Add Webhook

DevOps Use Case / Erstellen einer Pipeline

4. Pipeline starten

Update Webhook

Gitea will send POST requests with a specified content type to the target URL. Read more in the [webhooks guide](#).

Target URL *

HTTP Method

Secret

Trigger On:

Push Events
 All Events
 Custom Events...

Branch filter

Branch whitelist for push, branch creation and branch deletion events, specified as glob pattern. If empty or *, events for all branches are reported. See [github.com/gobwas/glob](#) documentation for syntax. Examples: master, {master, release*}.

Active
Information about triggered events will be sent to this webhook URL.

Recent Deliveries

2caba7c9-d4e4-48fa-8115-f04c1d1735bf 2020-07-03 14:47:26 UTC

DevOps Use Case / Erstellen einer Pipeline

5. Im Jenkins Interface: erfolgreicher Pipeline Durchlauf

The screenshot displays the Jenkins interface for a pipeline named 'DevOps_flow'. The main heading is 'Pipeline DevOps_flow'. Below it, there is a 'Recent Changes' section with a pencil icon. The 'Stage View' section shows a table of stage times and a build history entry.

Stage View

	Build	Deploy
Average stage times: (Average full run time: ~58s)	42s	15s
1 Jun 25 01:12 No Changes	42s	15s

Build History trend —

find X

1 Jun 24, 2020 11:12 PM

Atom feed for all Atom feed for failures

Permalinks

Praxis



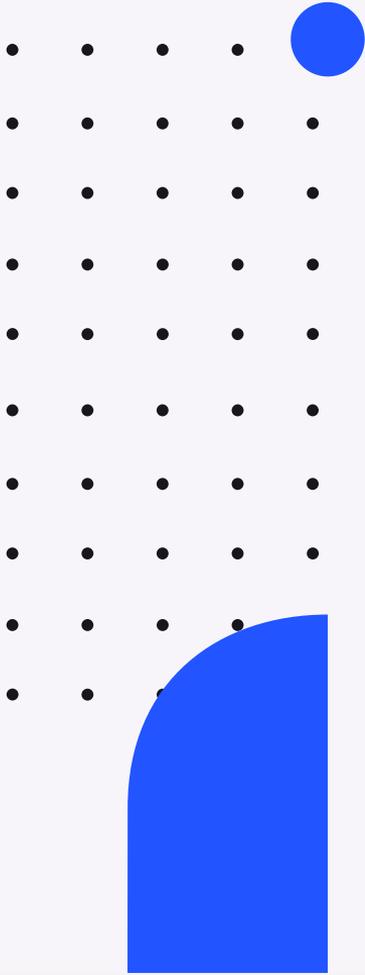
MQ Gateway Client Concentrator

Szenario: Der Kunde betreibt MQ Server unter Windows/X86 und plant MQ in zCX um Kommunikation mit MQ z/OS zu vereinfachen sowie Target Que Manager und Concentrator zu co-locaten.

PoC: Der Kunde betreibt MQ Gateways in 2 identitischen Farmen, wovon eine in zCX umgezogen und vergleichsweise auf Performanz getestet wird.

Ergebnis:

	Windows	zCX @ Mainframe
Measured peak loads	25-28% (of total system)	15-17% (of defined zCX-Instance)
Measured minimum utilization	10% (of total system)	2,7% (of defined zCX-Instance)
Average workload	12-15%	5-7%
RAM-Consumption	constant > 3 GB (incl. Windows)	ca. 1,2 GB (incl. zCX/Docker)
Max. possible Messages (extrapolated)	13.200 Msg/h	28.700 Msg/h



Praxis



Service Management Unite

Szenario: Der Kunde betreibt SMU derzeit unter Linux in z/VM . Da er mit SMU zufrieden ist, möchte er dies nun auch für z/OS vertesten.

Projekt: Der Kunde ist nicht überzeugt von der Hochverfügbarkeit und dem persistenten Speichern von zCX Docker Containern (Derzeit nur 2 Möglichkeiten: Neustart (1 Min Outage oder permanent Speicher für eine neue zCX Instanz reservieren). Daher wurde ein continuous availability Konzept auf Anwendungsebene am Beispiel von SMU verprobt.

Ergebnis: Mit Hilfe eines Load Balancers, DB2 Servers, 2 SMU zCX Environments und Event Dispatchers wurde ein continuous availability Betriebskonzept erzielt.

Praxis

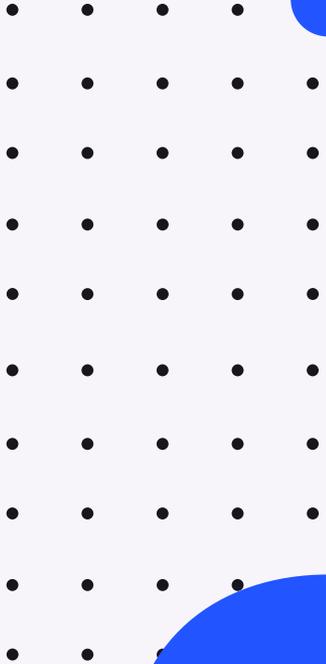


Watson Machine Learning for IBM Z

Szenario: Der Kunde hat aktuell das gesamte Training und Scoring seines Machine Learnings in der Cloud. Es werden Modelle trainiert und anschließend Bilder gescored/bewertet/ausgewertet. In Zukunft sollen allerdings Bilder mit einem hohen Schutzbedarf dazukommen (nicht Cloud).

PoC: Das Scoring der Machine Learning Applikation soll mit Hilfe von WMLz in zCX erfolgen. Die Bilder sollen dabei in DB2 auf z/OS gespeichert werden.

Ergebnis: Mit Hilfe einiger Anpassungen des WMLz Produkts konnten im PoC alle Modelle angewandt und Bilder gescored werden. Dies dauert wesentlich länger als in der Cloud, die Anwendung ist jedoch nicht zeitkritisch.



Praxis

Grafana

Szenario: Da zCX keinen Zugriff auf den Linux Kernel erlaubt monitored der Kunde seine zCX Umgebung mit Hilfe von Grafana.

Python Anwendungen

Szenario: Python Web Server einer Monitoring Applikation, die bisher unter RHEL 8 auf z/VM lief, erfolgreich umgezogen.

TPC-C OLTP Benchmark

Szenario: Der Kunde hat Applikation auf X86 die auf eine DB2 im z/OS zugreifen. Getestet wurden die möglichen Transaktionen pro Sekunde.

Ergebnis: zCX performte bei jedem Test mindestens 20% besser.

Danke

Maike Havemann
IBM Z Client Technical Specialist

—
maike.havemann@de.ibm.com
+49-174 1676826
[ibm.com](https://www.ibm.com)

