Modul 324 - DevOps

Woche 3

Lokales Entwickeln

Modul 324

1

Die Idee ist:

alle haben die gleiche Entwicklungsumgebung, selbst auf Produktion!



Modul 324 - DevOps

Verwendete Tools

- Docker Desktop
- Docker Compose
- VS Code
- Git oder Github Desktop



Repository klonen

- 1. Unter Code Das grüne Dropdown<> Code öffnen
- 2. Die SSH Url kopieren
- 3. oder Sofern GitHub Desktop installiert wurde **Open with GitHub Desktop** wählen.

■ C codingluke / m324-nginx-clone A	Q @
	ests 🕞 Actions 🖽 Projects 🔃 Security
⊙ ५ ☆	
☆ 0 stars 양 0 forks ⊙ 1 watching	양 1 Branch 📀 0 Tags -사- Activity
Private repository · Generated from co	odingluke/bbzbl-modul-324-nginx
l'main - l'	
Codingluke Update index.html	Local Codespaces
codingluke Update index.html	Local Codespaces
codingluke Update index.html .devcontainer .github	Local Codespaces
 codingluke Update index.html .devcontainer .github kamal 	Local Codespaces
 codingluke Update index.html .devcontainer .github kamal local-template 	Local Codespaces Local Codespaces E Clone ? HTTPS SSH GitHub CLI git@github.com:codingluke/m324-nginx-clone.
 codingluke Update index.html .devcontainer .github kamal local-template src 	Local Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespac
 codingluke Update index.html .devcontainer .github kamal local-template src torreform 	Local Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces Codespaces C

GitHub Desktop

Github Desktop ist ein Grafisches Tool um git repositories zu verwalten. In Github Desktop kann man mit einem GUI Commiten.

bbzbl-modul-324-nginx		۴	Current main	Brand	h v Push origin Last fetched just now 2 ↑ •
Changes 4 History		dev.E	Docker	file	\$.
4 changed files					@@ -24,7 +24,7 @@ CMD ["/usr/sbin/nginx", "-g", "daemon off;"]
.mise.toml	Đ				SHELL ["/bin/bash", "-c"]
			25	25	## Common toole
Z dau Daalaatila		/ /	27	20	 RUN apt-get -y update && apt-get -y install jq less openssl openssh-client gpg wget zip unzip a
					onf patch build-essential rustc libssl-dev libyaml-dev libreadline6-dev zlib1g-dev libgmp-dev 1
terraform/fast-deploy.sh					urses5-dev libffi-dev libgdbm6 libgdbm-dev libdb-dev uuid-devno-install-recommends
		×.		27	+ RUN apt-get -y update 66 apt-get -y install neovim jq less openssl openssh-client gpg wget zip p autoconf patch build-essential rustc libssl-dev libyaml-dev libreadline6-dev zlib1g-dev libgn v libncurses5-dev libffi-dev libgdbm6 libgdbm-dev libdb-dev uuid-devno-install-recommends
			29	29	## AWS CLI Apple Silicon
					@@ -53,7 +53,10 @@ RUN sh -c "\$(wget -0- https://github.com/deluan/zsh-in-docker/releases/down)
					&& curl −sS https://starship.rs/install.sh sh −s −yes \
					&& echo 'eval "\$(starship init bash)"' >> ~/.bashrc \
		11	55	55	$d\omega$ echo 'eval "\$(starship init zsh)"' >> ~/.zshrc \ - $\delta\delta$ echo 'eval "\$(~/ local/hin/mise activate zsh)"' >> ~/ zshrc \
		-	57		
		~	58		
				56	+ && echo 'eval "\$(~/.local/bin/mise activate zsh)"' >> ~/.zshrc
			59	57	, # Land/in
		ž		59	+ RUN git clone https://github.com/LazyVim/starter ~/.config/nvim \
Summary (required)		~		60	+ && rm -rf ~/.config/nvim/.git
				61	+ # && ~/.local/bin/mise plugins install neovim \
Description		1		62	+ # && ~/.local/bin/mise usealobal neovim@nightly \

Öffnen in VS Code

- 1. File -> Open folder
- 2. Das frisch geklonte repository auswählen
- 3. Es sollte ungefähr wie rechts aussehen



Modul 324 - DevOps

VS Code Plugins

Core

- Dev Containers
- Docker
- Remote SSH
- EditorConfig



Öffnen in Devcontainer

VS Code fragt automatisch nach, ob das Projekt im Container geöffnet werden soll sofern "Devcontainer" installiert wurde.

- "Reopen in Container" klicken und warten
- Nun wird der Container gebaut und gestartet. Das kann einige Minuten dauern!



Devcontainer start

- Wen auf "Connecting to Dev Container (Show Logs)" geklickt wird
- 2. erscheint folgender Log. Es zeigt wie das "Image" gebaut wird
- Unten rechts ist ersichtlich ob VS Code in einem Container geöffnet wird/wurde.

•••	$\leftarrow ~ ightarrow$ ul-324-nginx [Dev Container: Existing Docker Compose (Ext	
	.mise.toml ×	
└└── ── ── ── ── ── ── ── ── ── ── ── ──		
🔿 🛛 × 🌣 .mise.toml		
V BBZBL-MODUL-324-NGINX [DEV	/ CON	
	PROBLEMS OUTPUT TERMINAL PORTS Dev Containers $+ \vee$ \Rightarrow CACHED [nginx dev_container_auto_added_stage_label 2/10] RUN \Rightarrow CACHED [nginx dev_container_auto_added_stage_label 3/10] COPY [+] Building 154.9s (11/14) \Rightarrow Inginx internal] load build definition from Dockerfile-with-fei $\Rightarrow \Rightarrow$ transferring dockerfile: 2.78kB \Rightarrow [nginx internal] load metadata for docker.io/library/ubuntu:24 \Rightarrow [nginx internal] load metadata for docker.io/library/ubuntu:24 \Rightarrow [nginx internal] load metadata for docker.io/library/ubuntu:24 \Rightarrow [nginx internal] load dockerignore $\Rightarrow \Rightarrow$ transferring context: 28 \Rightarrow [nginx dev_container_auto_added_stage_label 1/10] FROM docker \Rightarrow [nginx dev_container_auto_added_stage_label 2/10] RUN dev_container_auto_added_stage_label 3/10] COPY \Rightarrow CACHED [nginx dev_container_auto_added_stage_label 3/10] COPY \Rightarrow CACHED [nginx dev_container_auto_added_stage_label 5/10] RUN apt-get \Rightarrow [nginx dev_container_auto_added_stage_label 5/10] RUN wapt-get \Rightarrow [nginx dev_container_auto_added_stage_label 5/10] RUN curl int \Rightarrow \Rightarrow # 100 19.5M 0 0 7273K 0 0:00:02 0 \Rightarrow \Rightarrow $\#$ \rightarrow ./COI \Rightarrow \Rightarrow $\#$ \Rightarrow (COI \Rightarrow \Rightarrow $\#$ \Rightarrow (COI) \Rightarrow \Rightarrow $\#$ \Rightarrow (COI)	<pre></pre>

Devcontainer Terminal

- 1. Mit + kann ein neues Terminal geöffnet werden, (z.B. zsh)
- 2. Nun existiert ein ubuntu Terminal im Container. (selbst unter Windows)
- 3. Die Dateien sind "gemountet" unter /workspace



Docker Compose

Infos unter Docker Compose

• Es verwendet das

.devcontainer/Dockerfile

- Momentan wird ./src zum Nginx gemountet
- ./local/.env Environment
 Variablen werden auch im Container hinzugefügt
- Der Port 3000 wird exposed.

```
# docker-compose.yml
services:
    nginx:
    build:
        context: .
        dockerfile: .devcontainer/Dockerfile
        container_name: devcontainer
    ports:
        - "3000:3000"
    env_file: ./local/.env
    volumes:
        # sync workspace
        - .:/workspace:cached
    ...
```

Installierte Tools im Container

Im Container sind folgendes Tool vorhanden

• mise

Optional (Auskommentiert):

- AWS Cli um nach AWS zu connecten
- Terraform um in AWS die Infrastruktur hochzufahren
- Kamal um ein Dockerfile zu deployen



Installiere Programmiersprachen mit mise

Java

mise use java@lts -g
openjdk version "24-loom" 2025-03-18
OpenJDK Runtime Environment (build 24-loom+4-42)
OpenJDK 64-Bit Server VM (build 24-loom+4-42, mixed mode, sharing)
mise java@24.0.0-loom+4-42 ✓ installed
mise /workspaces/bbzbl-modul-324-nginx/.mise.toml tools: java@24.0.0-loom+4-42

NodeJs

mise use node@lts -g
mise node@22.7.0 installed
mise /workspaces/bbzbl-modul-324-nginx/.mise.toml tools: node@22.7.0

Installiere Programmiersprachen im Dockerfile

Wir ein Container neu gestartet, müssen die Programmiersprachen neu installiert werden.

• Durch das

.devcontainer/Dockerfile können Sprachen beim erstellen vom Image installiert werden.

 So wird ermöglicht, dass alle Teammitglieder die gleiche Umgebung besitzen. • Hier ein Beispiel für nodejs

RUN mise use nodealts -g