Fullstackentwicklung mit MacOS

Bastian Nolte

February 27, 2019

1 Xcode

Xcode ist eine integrierte Entwicklungsumgebung für macOS mit einer Reihe von Softwareentwicklungswerkzeugen, die von Apple für die Entwicklung von Software für macOS, iOS, watchOS und tvOS entwickelt wurden.

Installiere Xcode über den AppStore von Apple.

1.1 Xcode command line tools

Xcode verfügt auch über einen Satz von Kommandozeilenwerkzeugen. Diese installierst Du am besten über das Terminal. Das Terminal kannst Du öffnen, indem Du ⌘+Leertaste drückst und dann `terminal` eingibst und ⏎ drückst.

Danach kann die Installation der `Xcode command line tools` wie folgt durchgeführt werden.

```
xcode-select --install
```

2 Paketmanager

MacOS bringt keinen eigenen Paketmanager mit, wie man ihn von anderen Unix-Derivaten kennt. Daher haben es sich verschiedene Projekte zur Aufgabe gemacht, diese Lücke zu füllen.

Die folgende Liste liefert einen Überblick aktueller Paketmanager im März 2019.

Mac Paketmanager	Pakete	Benötigt Rootrechte
Homebrew	4635	Nein
Homebrew Cask	4051	Nein
Nix package manager	15858	Nein
pkgsrc	18560	Ja
MacPorts	20572	Ja

Die Informationen wurden von der Seite Slant... What are the best Mac package managers? bezogen.

2.1 Homebrew installieren

Homebrew kann einfach über das Terminal installiert werden.

Die Installation erfolgt danach durch Eingabe des folgenden Kommandos im soeben geöffneten Terminal:

```
/usr/bin/ruby -e \
"$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"
```

Homebrew-Cask kann wie folgt verfügbar gemacht werden

```
brew tap caskroom/cask
```

2.2 Die wichtigsten Homebrewkommandos

```
# Nach einem bestimmten Paket suchen
brew search <paketname>

# Das OpenSource-Paket mit dem Namen <paketname> installieren
brew install <paketname>

# Ein nicht OpenSource-Paket installieren,  z.B. kommerzielle MacOS-Apps.
brew cask install <paketname>

# Alle bereits installierten Pakete und deren mitinstallierte Abhängigkeiten
# in einer Baumansicht auflisten.
brew deps --tree --installed

# Alle installierten Pakete in einer flachen Ansicht auflisten
brew list && brew cask list

# Informationen zu einem bereits installierten Paket erhalten
brew info <paketname>
```

Weitere Informationen findest Du auf der Homepage des Homebrew-Projektes.

3 iTerm2

iTerm2 ist ein Open-Source-Ersatz für Apples Terminal. Es ist sehr anpassungsfähig und verfügt über viele nützliche Funktionen.

Du kannst es wie folgt installieren:

```
brew cask install iterm2
```

Du startest iTerm2 indem Du ⌘+Leertaste drückst und dann `iterm` eingibst und ⏎ drückst.

3.1 Tastenkombinationen anpassen

Wahrscheinlich kennst Du die Tastenkombinationen mit denen Du wortweise (⌥) vor- und zurückspringen, beziehungsweise zum Anfang und Ende der Zeile (⌘) navigieren kannst.

Auch iTerm kannst Du so einstellen. Öffne dazu die iTerm2-Einstellungen (⌘+,) und navigiere dann zu `Profile` > `Schlüssel` und klicke auf das Symbol + um eine neue Tastenkombination hinzufügen.

Keyboard Shortcut	Action	Send
⌘←	Send escape sequence	OH
⌘→	Send escape sequence	OF
⌥←	Send escape sequence	b
⌥→	Send escape sequence	f

Nun kannst Du wortweise vor- und zurück, sowie an den Anfang und das Ende der Zeile navigieren.

4 Z-Shell

Die Z-Shell, die Du vielleicht unter dem Namen zsh kennst, ist eine Unix-Shell, die auf der Standard-Shell für macOS aufbaut und diese um weitere nützliche Funktionen erweitert.

Installieren kannst Du die zsh durch Eingabe folgenden Befehls:

```
brew install zsh
```

Du solltest ein Framework mit der zsh installieren, da es die Konfiguration und den Einsatz von Plugins und Themes deutlich erleichtert. Es gibt zwei besonders populäre Frameworks, Oh My Zsh und Prezto. Bitte beachte, dass Du nur ein Framework und nicht beide installieren solltest, da Du sonst mit Problemen rechnen musst.

4.1 Oh My Zsh

Ich habe mich dafür entschieden `Oh My Zsh` einzusetzen. Um es zu installieren, gibst Du folgenden Befehl in Dein Terminal ein.

```
sh -c "$(curl -fsSL https://raw.githubusercontent.com/robbyrussell\
/oh-my-zsh/master/tools/install.sh)"
chsh -s $(which zsh) # Setzt die zsh als Standardshell.
```

Ein Upgrade kannst Du mit dem Kommando `upgrade_oh_my_zsh` durchführen. Starte `iTerm2` neu um zsh zu aktivieren.

4.2 Oh My Zsh anpassen

Die Einstellungen von `Oh My Zsh` nach der Installation sind schon recht brauchbar. Es bietet Dir allerdings diverse Möglichkeiten sein Aussehen und Verhalten an Deine Anforderungen anzupassen. Wie das funktioniert kannst Du im Wiki nachlesen.

Wir wollen für den Anfang einige Plugins installieren. Öffne hierzu die Datei `~/.zshrc` und finde die Zeile die mit `plugins=` beginnt und trage Plugins wie folgt ein:

```
plugins=(
  brew
  colored-man
  colorize
  docker
  git
  github
  history
  mvn
  ng
  node
  npm
  osx
  zsh-syntax-highlighting
)
```

Zudem könnte es sich als praktisch erweisen in der Eingabehistorie weiter zurückblicken zu können. Dies stellt man wie durch folgende Zeilen ein.

```
export HISTSIZE=10000
export HISTFILESIZE=10000
```

Die Änderungen werden wirksam sobald Du iTerm neu gestartet hast.

5 Secure Shell

Secure Shell oder SSH bezeichnet sowohl ein Netzwerkprotokoll als auch entsprechende Programme, mit deren Hilfe man auf eine sichere Art und Weise eine verschlüsselte Netzwerkverbindung mit einem entfernten Gerät herstellen kann. Häufig wird diese Methode verwendet, um lokal eine entfernte Kommandozeile verfügbar zu machen, das heißt, auf einer lokalen Konsole werden die Ausgaben der entfernten Konsole ausgegeben und die lokalen Tastatureingaben werden an den entfernten Rechner gesendet. Genutzt werden kann dies beispielsweise zur Fernwartung eines in einem entfernten Rechenzentrum stehenden Servers.[1]

5.1 Einen SSH-Schlüssel erzeugen

Ein SSH-Schlüssel besteht aus einem Paar von Dateien. Eine davon beinhaltet den privaten Schlüssel, der niemals an Dritte weitergegeben werden sollte. Die andere beinhaltet den öffentlichen Schlüssel. Diesen öffentlichen Schlüssel kannst Du dazu verwenden Dich an Systemen anzumelden ohne Benutzername und Passwort eingeben zu müssen.

Ein Schlüsselpaar kann wie folgt erzeugt werden:

```
ssh-keygen -t rsa
```

5.2 Den öffentlichen Schlüssel in die Zwischenablage kopieren

```
pbcopy < ~/.ssh/id_rsa.pub
```

5.3 Den öffenlichen Schlüssel übertragen

Überträgt man den öffentlichen SSH-Schlüssel auf einen anderen Computer, z.B. einen Server und hinterlegt diesen dort, dann kann man sich künftig an diesen Server anmelden, ohne Benutzernamen und Passwort eingeben zu müssen.

Der Schlüssel kann mit folgendem Befehl übertragen und hinterlegt werden:

```
# ssh-copy-id -i ~/.ssh/key_rsa.pub <benutzername>@<servername oder ip>, z.B.
ssh-copy-id -i ~/.ssh/key_rsa.pub musterfrau@webserver.example.com
```

5.4 SSH-Schlüssel, Phassphrasen und der MacOS-Schlüsselbund

Ich empfehle Dir Deine SSH-Schlüssel mit einer Passphrase zu versehen. Wenn Du darauf verzichtest, kann sich jede Person, die in den Besitz Deines privaten Schlüssels gelangt, mit Deinem Benutzeraccount anmelden und das an allen Systemen auf denen Du Deinen Schlüssel hinterlegt hast.

Normalerweise müsstest Du dann aber bei jeder Verwendung des Schlüssels Deine Passphrase eingeben. Dies kann schnell anstrengend werden oder - zum Beispiel bei der automatisierten Kommunikation zwischen zwei Computern sogar unmöglich sein.

Dies kannst Du vermeiden, indem Du Deine privaten Schlüsselidentitäten dem SSH-Authentifizierungsagenten hinzufügst und die Passphrasen im Schlüsselbund Deines MacOS-Benutzers hinterlegst.

```
# ssd-add fügt Deine SSH-Schlüsselidentitäten
# dem SSH authentication agent hinzu
# -K sorgt dafür, dass Deine Passphrases in Deinem Schlüsselbund hinterlegt werden.
ssh-add -K
```

SSH-Schlüssel kannst Du auch nachträglich mit eine Passphrase versehen.

```
ssh-keygen -p
```

6 Git

Git ist eine freie Software zur verteilten Versionsverwaltung von Dateien.

Installiere Git durch Eingabe des folgenden Kommandozeilenbefehls.

```
brew install git
```

Erzeuge eine `.gitignore`-Datei in Deinem Nutzerverzeichnis `~` die Du global nutzen wirst. Diese könnte zum Beispiel, wie folgt aussehen:

```
# Folder view configuration files
.DS_Store
Desktop.ini

# Thumbnail cache files
._*
Thumbs.db

# Files that might appear on external disks
.Spotlight-V100
.Trashes

# Java, IntelliJ, Eclipse, node_modules....
# TODO...
node_modules
venv
.sass-cache
```

Registriere die Datei in Deiner globalen Git-Konfiguration, um die Einstellungen wirksam zu machen.

```
git config --global core.excludesfile ~/.gitignore
```

6.1 SSH-Schlüssel hinterlegen

Der soeben erzeugte öffentliche SSH-Schlüssel sollte im Bitbucket hinterlegt werden, um den Transport der Daten über das SSH-Protokoll zu ermöglichen.

https://git.css.ch/plugins/servlet/ssh/account/keys

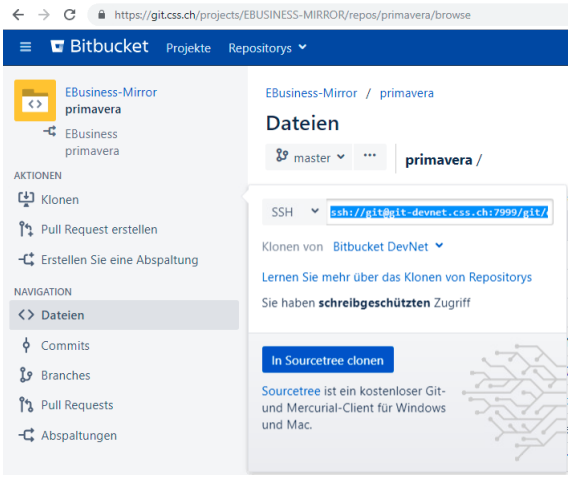
Dezidierte Informationen finden sich unter ...TODO...

6.2 Die GIT-Repositories im DEVNET verwenden

Nicht alle GIT-Repositories stehen im DEVNET zur Verfügung. Informationen ... finden sich im ....

Zu dem gespiegelten Projekten findet sich jeweils ein Spiegelserver-Projekt auf dem Bitbucket-server. Für EBusiness wäre das entsprechend EBUSINESS-MIRROR.

Klickt man auch der Bitbucketseite auf die Aktion `Klonen` so kann man nun im sich öffnenden Dialog die Auswahl "Klonen von" auf `Bitbucket DevNet` umstellen.



Für das Projekt `primavera` ergibt sich dann die git-url
`ssh://git@git-devnet.css.ch:7999/git/ebusiness-mirror/primavera.git`. Das Klonen erfolgt wie gewohnt mit dem Befehl
`git clone ssh://git@git-devnet.css.ch:7999/git/ebusiness-mirror/primavera.git`

7 Maven

...

7.1 Maven installieren

```
brew install maven
```

7.2 Maven konfigurieren

Maven Zugriff auf Artifactory aus dem DevNet

- Den Block `settings.xml` auf der Confluenceseite kopieren
- Eine neue Datei `settings.xml` im Verzeichnis `/.m2` anlegen
- Den soeben kopierten Konfigurationsblock in die Datei kopieren und die Datei speichern.
- An JFrog-Artifactory im DEVNET unter der URL `https://artifactory-devnet.css.ch/artifactory/webapp/#/home` anmelden und das verschlüsselte Passwort ermitteln.
- Benutzerlogin (p-Nummer) und verschlüsseltes Passwort in der soeben erzeugten Datei settings.xml anpassen.

```
# sed -i.bak -E 's!(<username>)[^<]*(</username>)!\1<Deine Login ID>\2!g' ~/.m2/settings.xml, z
sed -i.bak -E 's!(<username>)[^<]*(</username>)!\1p123123\2!g' ~/.m2/settings.xml
```

8 NPM installieren und konfigurieren

```
brew install npm
```

Aktuell nicht weiter konfigurieren »> ṅpmrc registry=http://artifactory.css.ch/artifactory/api/npm/npm/ sass_binary_site=http://repo.iasrv.css.ch/tools/node_sass_bindings «<

9 Entwicklungsumgebung

- google-chrome - iterm2 - intellij-idea (ultimate) - oracle-jdk - node - visual-studio-code - angular-cli - docker - nordvpn - postman - Zsh, Oh-my-zsh installiert und konfiguriert (OHNE prezto) Siehe auch https://sourabhbajaj.com/mac-setup/iTerm/zsh.html

10 vim

Vim The Ultimate vimrc https://sourabhbajaj.com/mac-setup/Vim/

11 Kommandozeilenbefehle

Liste nützlicher Kommandozeilenbefehle: Schaue die Dokumentation zu einem Kommandozeilenbefehl an

```
man <kommandozeilenbefehl>
```

Aktuellen Nutzer ausgeben

```
whoami
```

Aktuellen Pfad (Verzeichnis) ausgeben

```
pwd
```

Wechsle in ein Verzeichnis

```
#cd <verzeichnisname>, z.B.
cd /tmp
# Wechsle in das Nutzerverzeichnis (Heimatverzeichnis)
cd \~
# Wechsle zurück in das vorherige Verzeichnis
cd -
```

Tip: Nutze <tab> und <tab><tab> für automatische Ergänzungen.

Verzeichnis erstellen

```
mkdir <verzeichnisname>
```

Datei oder Verzeichnis verschieben

```
mv <quelle> <ziel>
```

Datei oder Verzeichnis kopieren

```
cp <quelle> <ziel>
```

Zeige den Inhalt einer Datei an:

```
# cat <dateiname>, z.B.
cat /var/log/meinserver.log
```

Gebe die neuesten Inhalte einer Datei auf der Konsole aus

```
tail -f /var/log/messages
# Mit Unterstützung von Logrotate
tail -F /var/log/messages
```

Alle Vorkommnisse eines regulären Ausdruckes in einer Datei ersetzen (substituieren), wobei die ursprüngliche Zustand in einer Kopie erhalten bleibt.

```
# sed -i.bu 's/<vorher>/<nachher>/g' <dateiname>, z.B.
sed -i.bu 's/Text/Hund/g' ./test.txt
```

Finde alle Dateien mit bestimmten Merkmalen. Beispiel

```
# Dateien die in den letzten 10 Minuten geändert wurden
find . -mmin -10
```

Die Befehlshistorie anzeigen

```
history
```

Den letzen Befehl nochmals als root ausführen

```
sudo !!
```

Screen

Screensession aufzeichnen

References

[1] *Secure Shell*. Wikipedia, Die freie Enzyklopädie. 2004. URL: `https://de.wikipedia.org/wiki/Secure_Shell` (visited on 02/26/2019).