Documentatie SRE-Challenge

IDE:

Als IDE heb ik VS Code gebruikt. De reden hiervoor is dat ik de extensie integratie erg fijn vindt werken en al bekend ben met de IDE door school.

Ook heb ik toegang tot Git Bash in VS code, wat in mijn mening het best werkt.

VM omgeving:

Als VM omgeving heb ik Docker gebruikt.

Efficiënt:

Docker is erg efficiënt omdat het weinig systeem resources vraagt en het over het algemeen snel werkt.

Voorkennis:

Ik heb voor verschillende projecten Docker gebruikt, hierdoor had ik al wat voorkennis over hoe Docker werkt.

Extensies:

In Docker bestaan er extensies die een image kunnen scannen op zwakke plekken, voor een eerder project heb ik Snyk gebruikt, in dit geval heb ik dat ook gedaan.

minikube

als minikube omgeving heb ik ook Docker gebruikt.

Tijdens het lezen van de minikube documentatie las ik dat Docker ook gebruikt kan worden voor het draaien van een minikube container, aangezien ik het VM gedeelte van de opdracht ook heb gedaan in Docker leek het me een goed plan om dit deel ook in Docker te maken.

Het kostte aardig wat moeite om dit werkend te krijgen, maar ik vermoed dat ik te weinig resources gaf aan Docker waardoor mijn Minikube image niet wou opstarten.

gebruikte resources:
ik heb de documentatie gebruikt die op de minikube site staat, ook heb ik een YouTube video gebruikt waarin de maker stap voor stap uitlegt hoe hij zijn eigen applicatie laat draaien via minikube.

Link naar video:

https://www.youtube.com/watch?v=qFhzu7LolUU&t=531s

vulnerabilities:

Nadat ik de complete SRE omgeving werkend heb gekregen kon ik met Snyk een scan uitvoeren, hieruit kwam 1 kritische fout.

Deze fout geeft aan dat in de library ‘zlib’ een integer overflow zit. Wanneer de maximale waarde binnen het systeem wordt bereikt en een +1 wordt gegeven kan het zijn dat de integer naar de laagst mogelijke waarde gaat.

Voorbeeld: bij een 8 bit systeem kan de integer gaat van -128 tot +127, wanneer er een 1 wordt opgeteld bij die 127 zal het systeem weer terugspringen naar -128, vandaar wordt het dus ook een wraparound genoemd.

Ik heb verder in de python code gezocht naar zwakke plekke / slecht geschreven code.

Ik heb niets aangepast, maar heb wel commentaar geplaatst op de plekken waar de code verbeterd kon worden.

Zo stond de secret key gewoon in plaintext in de code.

Werden alle gebruikers opgehaald bij een poging tot inloggen

En werden alle wachtwoorden in plaintext vergeleken met elkaar.