Lab Orchestrator Zwischenbericht

Marco Schlicht

August 11, 2021

Lab Orchestrator: Das Projekt Lab Orchestrator soll es vereinfachen VMs zu starten und auf diese über einen Browser zuzugreifen. Dabei soll ein User mehrere VMs in einem Netzwerk starten können um von einer VM auf die anderen zugreifen zu können. Dies sollte unter anderem dem Training von IT Security Personal helfen. Dazu kommen noch weitere Features wie beispielsweise eine Anleitung die nebenbei erklärt welche Schritte man in den VMs machen soll.

Kurzfassung: Die Arbeit am Projekt ist in vier Phasen aufgeteilt. Zuerst das Erarbeiten von Wissen mit einem Proof of Concept. In dieser Phase wurden die Grundlagen über Docker, Kubernetes und KubeVirt erarbeitet welche notwendig sind um dieses Projekt durchzuführen. Danach wurde in einem Proof of Concept gezeigt, dass die Idee des Lab Orchestrators mit diesen Technologien möglich ist. In der zweiten Phase wurde ein Prototype erstellt, welcher die Konzepte des Proof of Concepts in einem Programm abbildet um mögliche Probleme zu erkennen und frühzeitig zu lösen. Diese beiden Teile sind bereits erledigt und aktuell befindet sich das Projekt am Anfang von Phase 3. Phase 3 und 4 sind jeweils eine Alpha und Beta Phase des Projekts, in welchem das eigentliche Projekt durchgeführt wird. Hier wird eine Library und eine API entwickelt.

Dokumentation: Parallel zur Arbeit am Projekt, wird eine Projekt-Dokumentation geführt. Diese umfasst zur Zeit 116 Seiten und enthält fast alle Ergebnisse die sich während der Arbeit ergeben haben. Die Dokumentation fängt an mit einer Einleitung, welche die Ziele dieses Projekts und die einzelnen Phasen erklärt. Anschließend wird in dem Kapitel Basics einige Grundlagen von Kubernetes und weiteren Tools erklärt. Dieses Kapitel wird voraussichtlich nachträglich noch um Grundlagen von Docker ergänzt sobald eine Möglichkeit hinzugefügt wird Docker-Container statt Virtuellen Maschinen zu nutzen. Anschließend gibt es ein Kapitel welches zeigt wie die Development-Umgebung aufgesetzt werden kann. Dieses Kapitel wird am Ende des Projekts erweitert um eine Anleitung wie man das Projekt in einer Produktiv-Umgebung installiert. Danach folgen die Kapitel 4 “Proof of Concept” und Kapitel 5 “Prototype”. In dem “Proof of Concept” Kapitel wird auch anschließend noch ein Unterkapitel zu Docker ergänzt. Kapitel 5 ist fertig. Die weiteren Kapitel stehen noch nicht fest, sollen sich aber an den Phasen des Projekts orientieren. Die Dokumentation ist verfügbar unter MIT Lizenz und kann hier gefunden werden: github.com/LabOrchestrator/LabOrchestrator-Documentation[1].

Phase 1: Das erste Ziel von Phase 1 waren notwendiges Wissen erarbeiten. Grundwissen in Docker war bereits vorhanden deshalb wurde sich hier hauptsächlich Kubernetes angeschaut. Dabei lag ein besonderer Fokus auf der Kubernetes Erweiterung KubeVirt, welche es ermöglicht VMs in Kubernetes zu starten. Dazu wurde sich noch ttyd angeschaut, ein Tool mit welchem im Browser auf ein Terminal in einer VM oder einem Container zugegriffen werden kann und noVNC, ein Tool mit welchem im Browser auf VNC einer VM oder eines Containers zugegriffen werden kann. noVNC braucht anders als “normale” VNC Programme einen Websocket anstatt eines TCP Sockets. Praktischer weise bietet KubeVirt out of the box einen Websocket zu dem VNC der VMs an.

Die Ziele im zweiten Schritt der Phase 1 waren eine VM und ein Docker Image in Kubernetes über das Terminal und VNC im Web bereit zustellen. Dabei sollten in einem Lab mehrere VMs gestartet werden können und untereinander erreichbar sein, jedoch soll der Zugriff auf VMs in anderen Labs nicht möglich sein. Die Kombination aus KubeVirt, ttyd und noVNC wurde im Proof of Concept genutzt um das erste Ziel zu erfüllen. Für die Erreichung der Trennung der Labs wurde dann Kubernetes Network Policies genutzt.

Die grundlegenden Ziele Aneignung von Wissen und der Proof of Concept wurden in dieser Phase erfüllt. Hier wurden mehrere VMs pro Lab und mehrere Labs gestartet und es konnte sich über ttyd und noVNC damit verbunden werden. Die Labs waren untereinander nicht zu erreichen. Was fehlt sind die Docker Container. Es wurde sich nicht angeschaut wie das ganze mit Docker Containern erreicht werden kann sondern lediglich über VMs mit KubeVirt. Da allerdings VMs funktional ausreichend sind wurde sich darauf konzentriert. Docker wird am Ende noch hinzugefügt wenn noch Zeit übrig ist.

Phase 2: Das Ziel in Phase 2 war es einen Prototype zu programmieren, welcher die Konzepte aus Phase 1 nutzt und in einer API zur Verfügung stellt. Dabei sollte ein Docker Container erstellt werden, welcher diese App in Kubernetes deployed. Am Ende soll diese API über Kubernetes abrufbar sein, das Erstellen von Labs und VMs, den Zugriff auf diese über ttyd und noVNC ermöglichen und einen Layer für User Authentication hinzufügen. Das Deployment in dem Cluster wurde über Docker im Schritt 1 gemacht. Diese App greift innerhalb von Kubernetes auf die Kubernetes API zu. Dafür wurde im zweiten Schritt ein Service Account mit passenden RBAC Rules genutzt und die Kubernetes API angeschaut. Im dritten Schritt wurde in der App die Funktionalität hinzugefügt um VMIs über die API zu bekommen, neue VMIs zu starten und Authorisation zu dem VNC Zugriff hinzuzufügen. Für die API wurde hier Flask genutzt. Der VNC Zugriff hat standardmäßig keine Authorisation und damit wir unser User Management darauf anwenden können musste ein Websocket Proxy entwickelt werden, welcher anschließend in die App eingebunden wurde. Dieser Websocket Proxy liest einen Token aus dem Pfad welchen man angibt und prüft ob der damit Autorisierte User auch die Rechte hat auf den VNC Websocket dieser VM zuzugreifen. Diese Methode was notwendig um über noVNC die Autorisierungs-Daten übertragen zu können. Anschließend wurde in Schritt 4 der Prototype refactored und User Support hinzugefügt.

Das Resultat ist eine API über welche Labs erstellt, gestartet und gelöscht werden können. Dabei gibt es ein einfaches User Management System in welchem User verschiedene Rechte haben können. Beispielsweise können Labs von allen Usern gestartet aber nur von Admins erstellt werden. Der Zugriff auf VNC wird über den Proxy geleitet und durch JWT Tokens wird der Zugriff geschützt. Diese beiden Ziele sind damit erreicht. Ein Ziel, nämlich der Zugriff auf ttyd, wurde nicht im Prototype umgesetzt, sondern wird erst in der Alpha Phase hinzugefügt. Das hatte alleinig den Grund, dass sich die doppelte Arbeit gespart werden wollte. ttyd nutzt auch Websockets und kann damit ebenfalls mittels des Websocket-Proxies genutzt werden.

Phase 3 und 4: Als nächstes steht die Phase 3 und 4 an, in welcher wir die Erfahrungen aus dem Prototype nutzen werden, um einiges in eine Library zu abstrahieren und eine REST API bereit zu stellen.

[1] https://github.com/LabOrchestrator/LabOrchestrator-Documentation