Redesigning and Implementing the Bootstrap of Large Scale Kubernetes Enterprise Infrastructure through Automated Self Contained CLI

Project 2b (T3_2000)

in the context of the examination for the

Bachelor of Science (B.Sc.)

in Computer Science (Informatik)

at the Cooperative State University Baden-Württemberg Karlsruhe

by

Yannik Schiebelhut

Submission Date:	September 19, 2022
Processing period:	July 4, 2022 - September 19, 2022
Matriculation number, Course:	3354235, TINF20B1
Training institution:	SAP SE Dietmar-Hopp-Allee 16 69190 Walldorf, Deutschland
Supervisor of the training institution:	Samed Guener
Appraiser of the Cooperative State University:	Prof. Dr. Johannes Freudenmann

Eidesstattliche Erklärung

Ich versichere hiermit, dass ich meine Project 2b (T3_2000) mit dem Thema:

> *Redesigning and Implementing the Bootstrap of Large Scale Kubernetes Enterprise Infrastructure through Automated Self Contained CLI*

gemäß § 5 der „Studien- und Prüfungsordnung DHBW Technik“ vom 29. September 2017 selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Arbeit wurde bisher keiner anderen Prüfungsbehörde vorgelegt und auch nicht veröffentlicht.

Ich versichere zudem, dass die eingereichte elektronische Fassung mit der gedruckten Fassung übereinstimmt.

Karlsruhe, den 15. August 2022

Schiebelhut, Yannik

Sperrvermerk

Die nachfolgende Arbeit enthält vertrauliche Daten der:

SAP SE
Dietmar-Hopp-Allee 16
69190 Walldorf, Deutschland

Der Inhalt dieser Arbeit darf weder als Ganzes noch in Auszügen Personen außerhalb des Prüfungsprozesses und des Evaluationsverfahrens zugänglich gemacht werden, sofern keine anderslautende Genehmigung vom Dualen Partner vorliegt.

Abstract

- Deutsch -

Um unternehmensübergreifende Prozesse auf ein neues Niveau zu heben, arbeitet SAP an der Entwicklung von *Cross-Company Workflow Collaboration*. Die dort anfallenden, geteilten Prozessdaten sollen in *Signavio Process Intelligence* importiert werden, um sie dort mittels Process Mining zu analysieren und in Key Performance Indicators darzustellen. Da bei Cross-Company Workflow die *SAP Blockchain* zur Datenhaltung verwendet wird, stellt die Aufbereitung der Daten für eine Analyse in Signavio jedoch eine Herausforderung dar.

Im Rahmen dieser Projektarbeit werden zunächst diese Schwierigkeiten herausgearbeitet. Anschließend wird sich mit der Konzeptionierung eines Datenintegrationsprozesses befasst. Hierzu wird zum einen erörtert, wo die Transformation der Daten aus Cross-Company Workflow in das angestrebte Format durchgeführt werden sollte. Andererseits werden auch verschiedene Transportwege beleuchtet, über die die aufbereiteten Daten in Signavio Process Intelligence bereitgestellt werden sollen. Auf Basis des erarbeiteten Konzepts wird ein Prototyp der Datenintegration implementiert.

Abstract

- English -

In order to raise cross company processes to a new level SAP is working on the development of Cross-Company Workflow. The shared process data is to be imported into *Signavio Process Intelligence* to be analyzed with process mining and to be presented in Key Performance Indicators. Since Cross-Company Workflow uses the *SAP Blockchain* to store data the preparation of the data for analysis within Signavio is a challenge.

In the context of this report, these difficulties will first be discussed. It then deals with the conceptual design of a data integration process. For this purpose, on the one hand, it is discussed where the conversion of the data from Cross-Company Workflow into the target format should be carried out. On the other hand, different transport routes are discussed, through which the processed data should be made available in Signavio Process Intelligence. Based on the developed concept, a prototype of the data integration is implemented.

Contents

List of Abbreviations

ACL Access Control List
API Application Programming Interface
AWS Amazon Web Services
CICD Continuous Integration and Continuous Delivery
CLI command-line interface
EKS Elastic Kubernetes Service
HCL HashiCorp Configuration Language
IaaS Infrastructure as a Service
IaC Infrastructure as Code
JSON JavaScript Object Notation
S3 Simple Storage Service
VCS Version Control System

List of Figures

List of Code Listings

1 Introduction

2 Fundamentals

2.1 Go

Go (also known as Golang) is an open source programming language that was started at Google in 2007 and initially launched in 2009. The language was designed to face engineering challenges at Google with the goal to make it "easy to build simple, reliable and efficient software". [1, 2] By now the compiled and statically typed language [3] is widely used and the way it approaches network concurrency and software engineering has influenced other languages to a noticeable extent. [1] Through its structure go supports programming on various levels of abstraction. For instance, one can embed Assembler or C code into a Go program or on the other hand combine groups of components into bigger, more complex components to realize abstract design patterns. [4] Nowadays, Go is a popular choice for everything related to DevOps and therefor also for the development of command line tools. [5]

Go also tries to provide its own, official solutions for common tasks in software development. When installing Go, it comes packed alongside with a formatter (which shall ensure uniform styling across all programs written in Go) and an included suite for unit testing, just to name a few examples. Furthermore, Go supports generating documentation based on comments in the source code; much alike JavaDoc. Go features an extensive standard library which depicts a good starting point for developing your own applications. In case one wants to include a third party library, this can be done via the *go get* command. It fetches the necessary resources (usually directly from a Version Control System (VCS)) and saves the modules as a dependency in the *go.mod* file. (cite go tour and go docs)

2.1.1 Cobra

Cobra is an open source library for Go. Its aim is to provide developers with an easy way to create modern command-line interface (CLI) applications. The cobra library is being used by noticeable projects like the CLIs for Kubernetes or for GitHub. The idea

behind Cobra's intended command schema is that commands of a well constructed CLI should read like sentences. This way, new users that are familiar with CLIs in general quickly feel native because interacting with the CLI feels more natural. In this approach, a command represents a certain action that the CLI can perform. This action than take arguments and flags to further specify on which objects and in which way the command should take action. With Cobra, one can also easily create nested subcommands. This means that a before mentioned command can also be divided into multiple sub-actions to enable detailed handling of complex actions. Further, benefits of Cobra are, among others, the automated generation of autocomplete for the most common shells as well as the ability to automatically create man pages. [6, 7]

2.2 Terraform

Modern enterprise infrastructure for software development usually makes use of cloud computing to dynamically adapt infrastructure to the fast-paced world of agile development. These clusters are usually distributed between multiple cloud services, each of which use their own individual Application Programming Interfaces (APIs) to configure their platform. Terraform is a tool that tries to minimize the effort needed to deploy these infrastructures in the long run by automating resource management as far as possible. It allows to uniformly describe the target infrastructure in an easy to learn, machine-readable definition language and automatically takes care of deploying this infrastructure at the individual Infrastructure as a Service (IaaS) providers. This approach is also known as Infrastructure as Code (IaC). With Terraform, it is also possible to save provisioned infrastructure setups as a Terraform configuration to reuse them at a later point in time or to arbitrarily extend and adapt the configuration. For configuration, either JavaScript Object Notation (JSON) or HashiCorp Configuration Language (HCL) can be used, with HCL being the preferred way by the developing company HashiCorp because of its advanced features compared to JSON. Terraform can only unleash its full potential because of cooperations with all major software and hardware vendors and providers. HashiCorp partners with over 160 companies and services the most noticeable ones being:

- AWS,
- Atlassian,

- Cloudflare,
- Google,
- Microsoft, and
- Oracle.

The most common use cases for Terraform include general IaC, managing Kubernetes (section 2.3), multi-cloud deployment, management of network infrastructures, and management of virtual machine images. Terraform additionally integrates tightly with other HashiCorp services like Vault (section 2.7).

2.3 Kubernetes

Kubernetes (often short: k8s) is an open source solution to ease up and automate management of container based services. While doing so, it follows a declarative paradigm. This means that the users just needs to describe the desired state – for example through the use of configuration files or via the Kubernetes CLI – and Kubernetes determines the steps by itself which are necessary to reach and maintain this state. Kubernetes also enables users to dynamically scale their applications and services. This means that the amount of resources, that are dedicated to an application, is adapted during runtime dependent, for example, on the current number of users. Furthermore, Kubernetes can perform load balancing and redundancy between different instances of the same service. [8, 9]

this sounds ugly

One instance of a Kubernetes system is called a cluster. A Cluster is composed of multiple nodes (which usually are virtual machines or physical servers) which run the actual applications. An Application runs inside some kind of container, internally called Pod. The interaction with a cluster is managed by the so-called *Kubernetes Master*. It is a central controlling unit. The user actually never interacts with the nodes themselves directly. [10]

2.4 Gardener

Even though there are tools that help with creating and updating single Kubernetes clusters, it is rather hard to manage many clusters. This is where Gardener comes into play. It is a Kubernetes native extension that manages Kubernetes clusters as a service, packing the performance to manage up to thousands of clusters. One key concept that is applied is so called self-hosting. This means that Kubernetes components are run inside of Kubernetes clusters and is done, because Kubernetes is the go-to way to easily manage software in the cloud. [11, 12, 13, 14]

Gardener's architecture is constructed much like Kubernetes itself although not individual Pods are managed but entire clusters. The root of all is the so-called *Garden Cluster*. It is the main interface for the Gardener administrator and hosts, among others, the Garden Cluster control plane, the Gardener dashboard, the Gardener API server, the Gardener controller manager and the Gardener scheduler. Then there are two other types of clusters, *Seed Clusters* and *Shoot Clusters*. One Seed Clusters manages the control planes (thus API server, scheduler, controller manager, machine controller etc.) of its Shoot Clusters. There is at least one Seed Cluster per IaaS provider and region. The Shoot Clusters control planes are deployed as Pods inside the Seed Cluster and can therefore be created with standard Kubernetes deployment and support rolling updates. Shoot Clusters only contain worker nodes and are the Kubernetes Clusters that actually become available to the end-user and can be ordered in a declarative way. The clusters created by Gardener are vanilla Kubernetes clusters independent of the underlying cloud provider. [13, 15] (More details of the described architecture can be seen in Figure 2.1.)

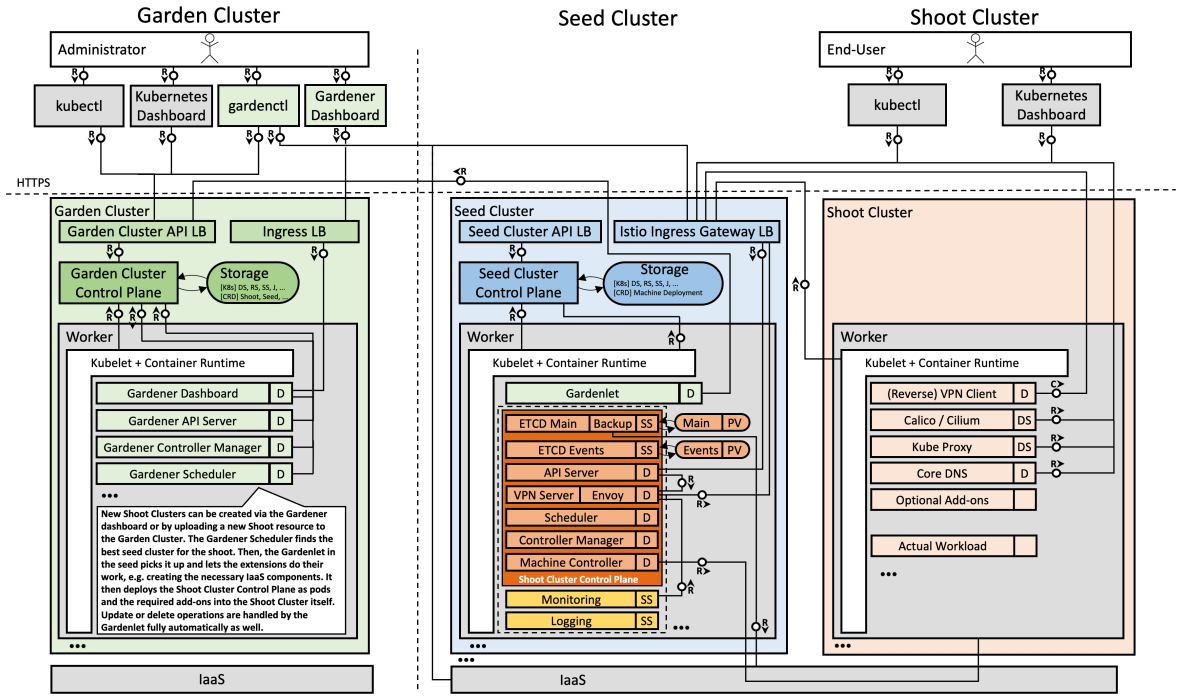

Figure 2.1: Gardener architecture [11]

2.5 AWS

AWS is a subsidiary company of Amazon that offers a cloud computing platform with various services, such as Simple Storage Service (S3) [16] or Elastic Kubernetes Service (EKS) [17]. Started in 2006, AWS nowadays runs data centers all over the world to provide scalable, reliable, and high performing services. [18, 19]

At the time of writing, AWS is the cloud provider with the biggest market share. [20] Also, most of the supervising department's clusters are currently being deployed on AWS. Because of these reasons and also due to constraints in time and complexity, the bootstrapping process worked out in this report is going to be narrowed down to deployment in an AWS environment.

2.6 Jenkins

Jenkins is a widely used open source automation server build with Java. It is a Continuous Integration and Continuous Delivery (CICD) tool aiming to save time by automating repetitive tasks like building projects, running test sets and deployment. Jenkins supports many plugins (over 1800) via its update center which give you the ability to integrate with most of the common tools in development and automate practicably any project. Jenkins can also be deployed on multiple machines to spread load and ensure quick and efficient operation. [21, 22, 23]

In context of this project, Jenkins will be used after the successful bootstrap to take care of running Terraform and performing the actual deployment.

2.7 Vault

Vault is an open source service with the primary task to provide a central control unit to manage and organize enterprise secrets. It encrypts secrets both at rest and in transit. Access to the secrets can be granted granular per user through the use of Access Control Lists (ACLs). Furthermore, vault audits access to the secrets. That means that it keeps a detailed log on whom accessed what secret at which point in time. If there was a security breach, where an unauthorized person got access to vault, this protocol can be used to tell, if a specific secret has been read by the attacker or if it is still safe to use.

Vault is designed to be highly pluggable. An instance is composed of *storage backends*, *audit log instances*, *authentication providers* as well as *secret backends*. Each of these can be impersonated by a variety of different components. This makes it possible to use different trusted authorities for attestation of identity. For example, among others LDAP, JWT, GitHub, and Radius can be used. An automated build service could very well use a different service to authenticate to vault than a human user.

Secrets and encryption are often the weak spot in applications. If a secret gets leaked and the leak stays unnoticed, attackers could gain long term access to a system. As a solution, Vault offers *dynamic secrets*. When a client requests the access credentials for a supported system, Vault creates a short-lived secret just for that specific client. Because the client is only accessing vault, it does not have to bother with key creation nor rotation

and an increased layer of security is added by not using secrets for an extended period of time. Also, if a dynamic secret gets leaked, this single secret can be revoked individually. If all clients accessing the resource used the same credentials, changing or blocking those could potentially cause an outage of the whole system.

When it comes to encryption, it can happen rather quickly that a single mistake compromises the security of the whole application. Because of this, vault offers encryption as a service. The idea is, that vault concentrates on the single task to handle credentials and encryption safely. The broad variety of applications have a different focus and are not developed with the necessary expertise to guarantee safe implementation of security measures. Vault, on the other hand, uses implementations that are audited by the open source community as well as independent experts. Those are then provided as a high level API to application developers. That way, the encryption process of data gets very easy while, at the same time, vault can handle the used encryption keys directly, and they are never actually sent to the application itself. [24]

During the bootstrap, this project is about, a technical user is created for Terraform operations. An access key is created for this user and then saved to vault. This way Jenkins then can obtain this access key and use it to run the actual cluster deployment with Terraform.

3 Evaluation and Future Work

Literaturverzeichnis

[1] Pike, R. *Using Go at Google - The Go Programming Language*. 08/2020. URL: https://go.dev/solutions/google/ (visited on 08/02/2022).

[2] *The Go programming language — GitHub*. URL: https://github.com/golang/go (visited on 08/02/2022).

[3] Chris, K. *What is Go? Golang Programming Language Meaning Explained*. 10/2021. URL: https://www.freecodecamp.org/news/what-is-go-programming-language/ (visited on 08/02/2022).

[4] Maurer, C. *Objektbasierte Programmierung mit Go*. Springer Vieweg, 2021.

[5] Van Sickle, M. *Go: Getting Started — Pluralsight*. 01/2020. URL: https://app.pluralsight.com/library/courses/getting-started-with-go/table-of-contents (visited on 08/02/2022).

[6] *A Commander for modern Go CLI interactions — GitHub*. URL: https://github.com/spf13/cobra (visited on 08/02/2022).

[7] *Cobra. Dev*. URL: https://cobra.dev/ (visited on 08/02/2022).

[8] Bloß, A. *Containerorchestrierung mit Kubernetes - Teil 4 — x-cellent technologies GmbH Blog*. 2019. URL: https://www.x-cellent.com/blog/containerorchestrierung-mit-kubernetes-teil-4/ (visited on 07/15/2021).

[9] *Was ist Kubernetes? — Kubernetes*. URL: https://kubernetes.io/de/docs/concepts/overview/what-is-kubernetes/%7B%5C#%7Dwas-bedeutet-kubernetes-k8s (visited on 07/15/2021).

[10] *Konzepte — Kubernetes*. URL: https://kubernetes.io/de/docs/concepts/%7B%5C_%7Dprint/ (visited on 07/15/2021).

[11] *gardener/architecture.md at master · gardener/gardener*. URL: https://github.com/gardener/gardener/blob/master/docs/concepts/architecture.md (visited on 08/15/2022).

[12] *Gardener - The Kubernetes Botanist — Kubernetes*. URL: https://kubernetes.io/blog/2018/05/17/gardener/ (visited on 08/15/2022).

[13] *Gardener Project Update — Kubernetes*. URL: https://kubernetes.io/blog/2019/12/02/gardener-project-update/ (visited on 08/15/2022).

[14] *Gardener*. URL: https://gardener.cloud/ (visited on 08/15/2022).

[15] *Was ist SAP Gardener?* URL: https://www.datacenter-insider.de/was-ist-sap-gardener-a-955022/ (visited on 08/15/2022).

[16] *Amazon Simple Storage Service S3 – Cloud Online-Speicher*. URL: https://aws.amazon.com/de/s3/ (visited on 08/15/2022).

[17] *Verwalteter Kubernetes-Service – Amazon EKS – Amazon Web Services*. URL: https://aws.amazon.com/de/eks/ (visited on 08/15/2022).

[18] *Amazon Web Services (AWS): Über uns — LinkedIn*. URL: https://www.linkedin.com/company/amazon-web-services/about/ (visited on 08/15/2022).

[19] *Was ist Amazon Web Services (AWS)? - Definition von WhatIs.com*. 2014. URL: https://www.computerweekly.com/de/definition/Amazon-Web-Services-AWS (visited on 08/15/2022).

[20] Kumar, R. *AWS Market Share 2022: How Far It Rules the Cloud Industry?* 05/2022. URL: https://www.wpoven.com/blog/aws-market-share/ (visited on 08/15/2022).

[21] *Jenkins*. URL: https://www.jenkins.io/ (visited on 08/15/2022).

[22] *Jenkins automation server — GitHub*. URL: https://github.com/jenkinsci/jenkins (visited on 08/15/2022).

[23] *What is CI/CD? — GitLab*. URL: https://about.gitlab.com/topics/ci-cd/ (visited on 08/15/2022).

[24] *Vault by HashiCorp*. URL: https://www.vaultproject.io/ (visited on 08/15/2022).