



Expertise
and insight
for the future

Julius Leppälä

Assisting Provet Cloud Users With Speech Recognition Technologies

Metropolia University of Applied Sciences

Master of Engineering

Information Technology

Master's Thesis

29 May 2020

PREFACE

I'm grateful that this study has been completed. It has been a difficult task to accomplish due to various reasons on the way. A lot of things have happened during this thesis: my daughter's birth, various technological changes in our company and the global COVID-19 pandemic.

The biggest learning takeaway for me is all new technology related to speech recognition, especially Dialogflow. It was very interesting to study how it works and how it could be implemented in existing applications. It was also the biggest challenge as I had zero knowledge about this technology field.

Additionally, I learnt many more things how Kubernetes works and how applications can be deployed there. During this study I found Helm which simplifies deployments to Kubernetes clusters. I'm going to use it in future projects on my career too.

I want to thank my instructor who gave me good ideas of what could be included in this study. Also, I want to thank my supervisor at the case company. With him we drafted the topic of this study and later conducted usability testing.

Special thanks go to my wife who gave birth to your lovely daughter during this study. Regardless of that she supported me and wanted to give me time to finish it.

Kirkkonummi, 29 May 2020
Julius Leppälä

Author Title Number of Pages Date	Julius Leppälä Assisting Provet Cloud Users With Speech Recognition Technologies 37 pages + 1 appendix 29 May 2020
Degree	Master of Engineering
Degree Programme	Information Technology
Instructor	Antti Koivumäki, Principal Lecturer
<p>The purpose of this study was to create a proof-of-concept application which integrates Google Assistant and the case company's application Provet Cloud. The main reason for this was to study whether it is possible and would be helpful for veterinary professionals to use speech recognition in their work.</p> <p>The study started as defining the scope. The goal was to build a solution where one can request data from Provet Cloud by talking to Google Assistant in English. The solution included one use case where a veterinary professional can ask incoming appointments on a specific date. Localization and other virtual assistants, like Amazon Alexa and Apple's Siri, were left out of the scope.</p> <p>After scope validation technical stack was decided. Studying technical stack required a lot of time, especially Dialogflow and Kubernetes. During solution development it became clear that adding an intention in Dialogflow and providing data for that requires a lot of work. It's even more complicated when one wants to build conversations that continue.</p> <p>Usability tests were carried with the supervisor and a veterinary professional. In addition, the developed code was reviewed by two developers focusing in the different areas of the proof-of-concept. One developer reviewed changes done in Provet Cloud and the other reviewed the code of Provet Flow.</p> <p>This study achieved its goal and integration between Google Assistant and Provet Cloud was possible. A user can ask his or her appointments on a specific date using Google Assistant. However, it became clear that the end user wouldn't use this on workdays at the veterinary clinic or hospital due to surrounding distractions, but at home to prepare for the next day.</p> <p>Future development consists of support for localization, more intents and publishing the Action. Additional development is needed to show the proof-of-concept, for example, at a business affair.</p>	
Keywords	speech, voice, recognition, assist, artificial intelligence

Kirjoittaja Otsikko	Julius Leppälä Assisting Provet Cloud Users With Speech Recognition Technologies
Sivumäärä Päivämäärä	37 sivua + 1 liite 29 toukokuuta 2020
Tutkintotaso	Master of Engineering
Tutkinto-ohjelma	Information Technology
Ohjaaja	Antti Koivumäki, Yliopettaja
<p>Tämän työn tarkoituksena oli luoda prototyyppi, joka yhdistää Google Assistantin ja asiakasyrityksen ohjelmiston, Provet Cloudin. Tarkoitus oli tutkia, olisiko eläinlääketieteen ammattilaisten mahdollista ja hyödyllistä käyttää äänentunnistusapuvälineitä heidän työssään.</p> <p>Tutkimus aloitettiin määrittämällä sen laajuus. Tarkoituksena oli mahdollistaa tiedon haku Provet Cloudista puhumalla Google Assistantille englanniksi. Prototyypissä oli oltava mahdollista kysyä tulevia ajanvarauksia tietynä päivänä. Lokalisaatio ja muut virtuaaliset avustajat jätettiin tämän työn ulkopuolelle.</p> <p>Seuraavaksi määritettiin käytettävät tekniset komponentit. Tarvittavien komponenttien opiskelu ja niiden päälle rakentaminen vei paljon aikaa, erityisesti Dialogflowin ja Kubernetesen opiskelu. Lisäksi työn edetessä tuli ilmi, että uuden käyttötapauksen lisääminen oli suhteellisen työlästä. Asia monimutkaistuu entisestään, jos niissä halutaan käyttää edellisen keskustelun kontekstia.</p> <p>Käytettävyydestä suoritettiin asiakasyrityksen ohjaajan ja eläinlääketieteen ammattilaisen kanssa. Lisäksi kaksi ohjelmoijaa katselmoivat projektin aikana syntyneen koodin keskittyen eri alueisiin. Yksi kehittäjä tarkasti Provet Cloudiin tehdyt muutokset ja toinen Provet Flowin koodin.</p> <p>Tämä työ saavutti päämääränsä eli integraatio Google Assistantin ja Provet Cloudin välillä onnistui. Käyttäjä pystyy kysymään Google Assistantilta, mitä ajanvarauksia hänelle on tulossa tietynä päivänä. Testauksessa tuli kuitenkin ilmi, että Google Assistantin käyttö on melkein mahdotonta eläinklinikalla tai -sairaalassa ympärillä olevan hälinän vuoksi. Sitä voisi kuitenkin käyttää kotona, kun valmistautuu seuraavaan työpäivään.</p> <p>Jatkokehitys koostuu lokalisatiotuesta, useammasta käyttötapauksesta ja tuotantojulkaisusta. Lisäkehitystä tarvitaan, jotta prototyyppiä voidaan esitellä jossakin, esimerkiksi messuilla.</p>	
Avainsanat	Puheentunnistus, äänentunnistus, avustus, tekoäly

Contents

Preface

Abstract

List of Figures

List of Abbreviations

1	Introduction	1
2	Project Specifications	2
2.1	Current State of the Source System	2
2.2	Requirements	3
3	Method and Material	5
3.1	Project Scope	5
3.2	Research Method and Design	5
3.3	Reliability and Validity	6
4	Required Applications and Tools	7
4.1	Google Assistant	7
4.2	Dialogflow	8
4.3	OAuth 2.0	9
4.4	Django	9
4.5	Docker Engine	10
4.6	Kubernetes	11
4.7	Helm	13
5	Solution Development	15
5.1	Requirements	15
5.2	Additional Development in Provet Cloud	15
5.3	Dialogflow Agent	16
5.3.1	Intents and Fulfillment	16
5.3.2	Deployment of the Action	18
5.4	Provet Flow	20
5.4.1	Technology Stack	20
5.4.2	User Workflow	21
5.4.3	GDPR	25
5.5	Deployment Workflow	26

5.6	Future Development	27
6	Results and Analysis	29
6.1	Usability Test Case	29
6.1.1	Test Results From a Colleague	29
6.1.2	Test Results From a Veterinary Professional	30
6.2	Technology Review	31
6.3	Code Review	31
7	Discussions and Conclusions	33

References

Appendices

Appendix 1. Instructions to activate Google Assistant integration with Provet Cloud

List of Figures

Figure 1. Provet Cloud's SaaS architecture.	2
Figure 2. Research design.	5
Figure 3. Google Home, Google Home Mini and Google Home Max speakers.	7
Figure 4. Comparison of virtual machines and containers.	10
Figure 5. Architectural diagram.	15
Figure 6. Intent for appointments.	17
Figure 7. Fulfillment.	18
Figure 8. Surface capabilities.	19
Figure 9. Technology stack.	21
Figure 10. User workflow.	21
Figure 11. Connect a Google account in Provet Cloud.	22
Figure 12. Google authentication.	22
Figure 13. Successful authentication with Google.	23
Figure 14. Registration.	23
Figure 15. Details about Google Assistant connection.	24
Figure 16. Revoke connection to Google Assistant.	25
Figure 17. CI/CD pipeline.	27
Figure 18. Assert statement without optimizations.	32
Figure 19. Assert statement with optimizations.	32

List of Abbreviations

AI	Artificial Intelligence.
API	Application Programming Interface.
AWS	Amazon Web Services. A global cloud computing platform.
DNS	Domain Name System.
EBS	Elastic Block Store. Volume management in AWS.
GDPR	General Data Protection Regulation.
JSON	JavaScript Object Notation.
MVC	Model-view-controller. Architectural pattern for programming.
ORM	Object-relational mapping. A programming technique to map types in object-oriented programming as database structure.
REST	A set of constraints for web services, abbreviated from Representational State Transfer.
RESTful API	A combination of two definitions above. Used to fetch and manipulate data in a system by doing web requests in agreed format, such as JSON.
REST API	Abbreviated from RESTful API.
SaaS	Software as a Service.
YAML	YAML Ain't Markup Language. Data-serialization language commonly used in configuration files.
URL	Uniform Resource Locator. Web address.

1 Introduction

During the last 10 years smart homes, applications and devices have come more and more popular. For example, nowadays it is possible to have a voice control for everything: lighting, music, doors etc. One can ask for calendar events from Google Assistant or one can switch on living room lights using Amazon Alexa.

The case company in this thesis, Finnish Net Solutions, is the creator of Provet Cloud which is a cloud-based practice management system for veterinary professionals. It works as a Software as a Service (SaaS) and can scale from one-man clinics to huge veterinary hospitals. Development was started in 2014 and the first customers started using the solution in 2015. Currently it has several thousand users all over the world.

The case company has always wanted to keep up with new technologies and innovations. A new idea was born few months ago: provide user assistance in Provet Cloud by a voice control. Harnessing voice control is a huge task so the company decided to start with a small proof-of-concept which also became the goal of this project: allow fetching data of incoming appointments by asking from Google Assistant.

This study describes all work done to bring a proof-of-concept to life. It starts from technical stack definition and deployment configurations and continues to testing, analyses and conclusions. The idea was also to use state-of-art technologies to run the proof-of-concept application.

The leading speech recognition technologies are currently Google Assistant, Amazon Alexa and Siri from Apple. Google was chosen as provider for speech recognition to narrow the scope of the study. The resulting application should be extendable to be used with Amazon Alexa and Siri too.

2 Project Specifications

This section describes current state of what already has been implemented and what still needs to be done in order to fulfill goal of this study.

2.1 Current State of the Source System

As mentioned in the first chapter, Provet Cloud is a SaaS system that can fit for veterinary clinics of multiple sizes. It's built as multitenant system, which means that each customer has a separate database with their individual data, but all share same codebase. Multi-tenancy is a very common approach today when building SaaS systems. [1]

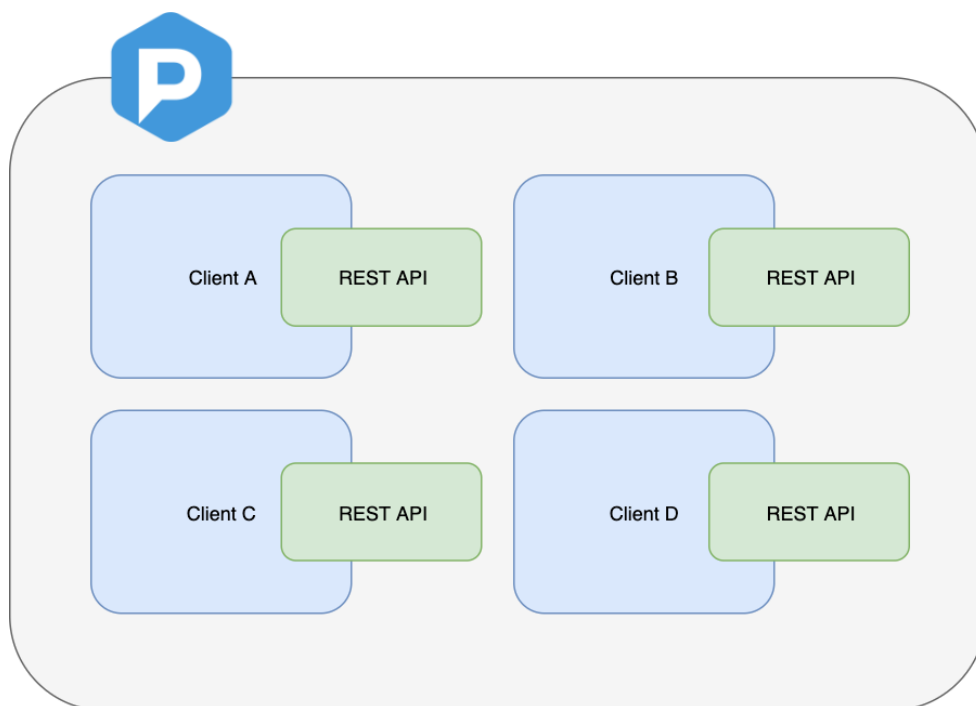


Figure 1. Provet Cloud's SaaS architecture.

Provet Cloud has a working Representational State Transfer Application Programming Interface (RESTful API, REST API) which allows users to query various types of data from the system. Users need a token created by their system's administrator in order to make authenticated web requests. An administrator can define permissions for tokens so it can be used to access only needed data and nothing else.

Currently there are no integrations to any voice control -related functions or applications in Provet Cloud, but its REST API is widely used for various applications, mainly to retrieve financial or treatment data to external systems.

2.2 Requirements

The main objective is to use voice commands in Google Assistant to retrieve data from Provet Cloud and build audible responses for the user. Communication must be only in one direction meaning that it's not needed to write data in Provet Cloud using Google Assistant – only reading is required.

Firstly, a major missing component is a service which provides integration between Provet Cloud and Google Assistant. The project name for this component is Provet Flow. It accepts requests from Google Assistant, identifies the connected Google account and routes requests forward to the correct Provet Cloud system. Provet Cloud's REST API returns response to Provet Flow, which is then parsed and returned to the client.

Secondly, a conversational action needs to be built for Google Assistant, which parses user request and sends it further. Google provides Dialogflow for this and it plays an important role in this study. Dialogflow will handle all processing regarding to speech recognition and sends request forward to Provet Flow.

Thirdly, Provet Cloud needs some changes to link a Google account into user account and send this information to Provet Flow. User must verify his or her Google account and link to their Provet Cloud account. Provet Flow must have a REST API endpoint to receive this information.

General Data Protection Regulation (GDPR) must be taken into account when building the solution. Google is a global company so any critical user data, like identification numbers, must not be exposed to Google Assistant. Provet Flow must parse responses from Provet Cloud in a way that they are safe to be sent to Google Assistant.

Localization was scoped out of this study and the developed application works only in English. Google Assistant supports multiple languages, currently 30 languages, and localization is also included in future development ideas of Provet Flow. For Provet Cloud purposes, the second most relevant language would be Swedish which is also supported by Google Assistant. [2]

3 Method and Material

This section describes project scope and research method and design.

3.1 Project Scope

Connecting Google Assistant, or other virtual assistant service, with other third-party software requires reliable components to handle communication between each other. Luckily the most vital part, speech recognition and natural language processing (NLP), is handled by Dialogflow and the focus can be targeted on the communication between the two software. This gives more possibility to concentrate on the development process and work on the desired use cases. NLP includes all features of machine learning that are being used in Dialogflow which actually handles speech recognition. Development of these kind of features were left out of this project.

Google Assistant was the only virtual assistant included in this study to narrow the scope and keep better focus. Technically other available assistants, like Amazon Alexa or Apple's Siri, could have been used too.

This study was narrowed to English language only. As said in chapter 2.2, Google Assistant supports multiple languages, but developing for multiple languages requires more effort and knowledge of the language itself.

3.2 Research Method and Design

This qualitative research includes more practical development than theory. It also requires a plenty of researching, studying and planning the proof-of-concept. A lot of studying was done online as there is much more material online about development on Google Assistant compared to written material.

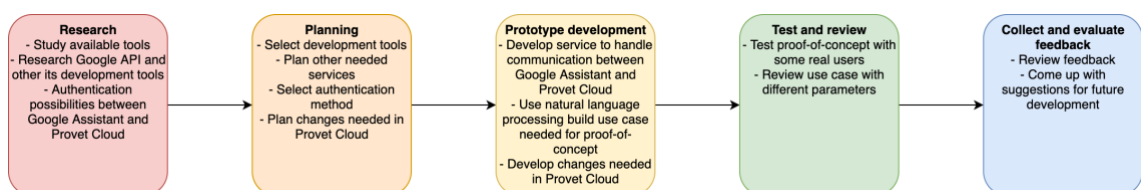


Figure 2. Research design.

3.3 Reliability and Validity

The evaluation of this thesis and solution was done during and after coding process. Validation included three types of testing: technical review, code review and usability testing. Feedback from different perspectives was wanted to ensure proper validation.

Usability testing was carried out with a veterinary professional actively using Provet Cloud. The goal was to gather good experience and feedback from a real user who is also working in the veterinary field.

Technical review and code review were done in-house with a couple of developers. They reviewed the code for Provet Flow and additions to Provet Cloud. More experienced developers from the case company were chosen to do this to get the most valuable feedback. Results on the review can be found in the Results and Analysis section.

Future development ideas based on these results can be found in the Conclusions section. The objective of this study was not a profitable software, but to test newer technologies and their integration to Provet Cloud.

4 Required Applications and Tools

This section describes all needed applications, tools and platforms to develop the proof-of-concept application. Requirements consist of physical devices, third-party software and the application stack.

4.1 Google Assistant

Google Assistant is a dialog-enabled virtual assistant used in mobile and smart home devices. At its core artificial intelligence and natural language processing are powering all the functions.

In July 2012 Google released feature for Google Search called Google Now. It helped users by delivering proactively information they may need. It was available as an application for Android and iOS. The feature is still available, but all references to “Now” were removed by October 2016.

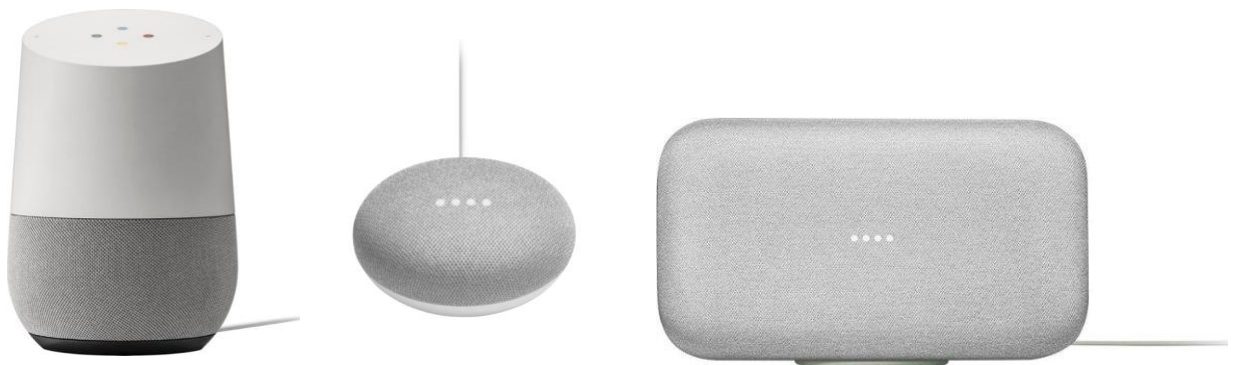


Figure 3. Google Home, Google Home Mini and Google Home Max speakers.

In the same year the first version of a voice-activated speaker, Google Home, was released. Google Assistant was included in the speaker and also in the messaging application Allo. During 2017 it was released for Android, iOS and Wear OS. After that Google has been implementing more and more features to enable wider interaction with smart devices via voice commands.

Currently Google has produced three types of their own speakers that have built-in Google Assistant. In addition to these, also many other speaker manufacturers, like Bose, have built their speaker variants with Google Assistant support.

4.2 Dialogflow

Dialogflow is a natural language processing engine which allows different contextual elements in dialogs. For example, it uses location and conversation history to achieve more targeted dialog with the user. Dialogflow supports follow-ups which means that it can continue the conversation by asking more targeted questions to provide more accurate answers.

Dialogflow, currently owned by Google, formerly api.ai, was founded in 2012. First version was released in 2014. In 2016 Google bought the company and engine was renamed to Dialogflow. [3] Currently it is one of the main tools used to develop applications for Google Assistant.

Intents are conversation building blocks in Dialogflow. They contain training phrases which are used to map a user query to the correct function. Inside an intent you can parse parameters out of the query. Intents can response to the user in the desired way. There two types of response: hard-coded text response or response from a server. For example, if response from the server is wanted, you must define fulfillment with a webhook.

Intents contain training phrases. More of them makes it easier to detect different variations of the user query. Dialogflow tries to detect defined parameters in all training phrases, but it can be fine-tuned if errors are detected.

Dialogflow contains many prebuilt parameters, but developer can create their own too. However, prebuilt ones provide good parsing and validation for the data. For example, Dialogflow can detect dates, numbers, addresses, names and much more. [4]

Fulfillment is a programmatic way to handle and answer to the user query. Dialogflow provides two options for this: webhook or code written in an inline editor. A webhook consists of a POST request to the target Uniform Resource Locator (URL). A request

contains all data from the user query, including phrase in written text and parsed parameters. The other option is to write code to the inline editor which uses Google's Cloud Functions and Firebase. [4]

4.3 OAuth 2.0

OAuth is an authentication protocol which allows websites and applications to share user data with each other without actually exposing authentication credentials, like passwords or tokens.

OAuth 1.0 was released in 2007. The original need was to allow secure access between Twitter and Magnolia APIs. A small group of people from different companies, including Google, gathered together and created an open standard for access delegation. OAuth 1.0 was born.

In October 2012 next version, OAuth 2.0, was released. It is not backwards compatible with the first release. OAuth 2.0 fixes security flaws and also enables more specific authorization flows for different platforms. [5]

4.4 Django

Django is an open source web application framework written in Python programming language. It was created in 2005 by two web programmers who decided to use Python to build web applications. In 2008 Django Software Foundation (DSF) was founded and development and maintenance of Django was transferred to it. [6]

Django works as model-view-controller architecture (MVC) with object-relational mapping (ORM) support. It has a large number of features to ease web application development, such as database migrations, form validation, a caching framework, templating engine and many more. Django also provides built-in authentication backend and supports many database backends. [7]

4.5 Docker Engine

Docker Engine is a software which runs containers containing various applications. It was created in 2010 by Solomon Hykes. First it was an internal project within company called dotCloud. Later on it was released as open source in 2013. [8]

Compared to virtual machines, Docker containers are more lightweight, flexible and portable. They don't require the same overhead to run a virtual machine but use the host's kernel to execute their application. Being lightweight, containers use more efficiently underlying hardware. When launching a container, one can allocate resource limits for it or allow usage without limits.

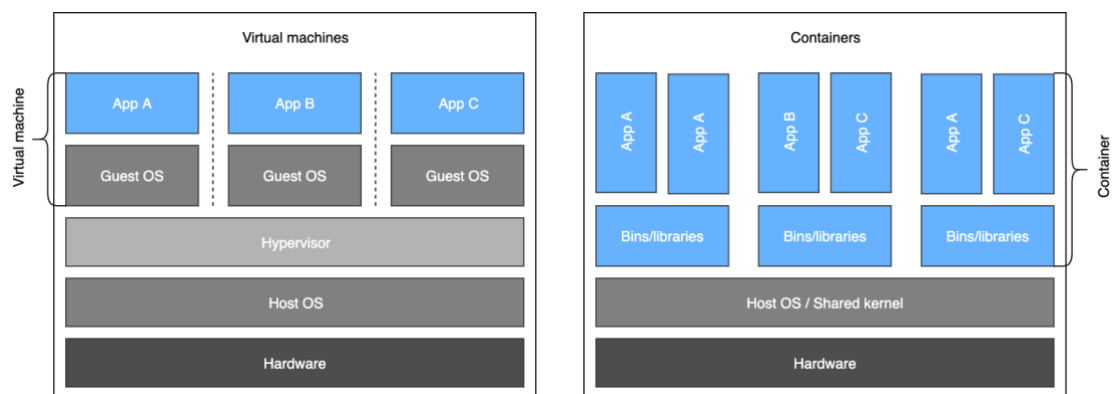


Figure 4. Comparison of virtual machines and containers.

Docker containers are run from images that developers can build. Images can base on desired Linux-distro and there exist some Windows base images too. A usual workflow is following: choose a base image, install dependencies, install or compile application and run the container.

Docker's power is portability: code, build and ship your application as a Docker image. Image can be run on any platform and it will always run in the same way. It enables developers to do their work on any platform. Same code will be run exactly in the same way on developer's computer as in the production server.

4.6 Kubernetes

The roots of Kubernetes are at Google. Before 2013 Google had built and been using its own cluster-management system called Borg. Borg is the predecessor of Kubernetes and it runs hundreds of thousands of jobs across Google's infrastructure. It allows Google to keep data centers at high utilization. [9]

In 2013 some employees at Google got an idea to build an open source container management system. The idea was to make it Borg-alike, but considering lessons learned when Borg was built. Another important factor was to make it open source in order to get fast feedback and contributions from hundreds of engineers around the world. [9]

Kubernetes was born and version was out in 2015. Already in 2016 over 830 contributors had been coding to make Kubernetes even better. It is used in thousands of companies to keep their cluster management simple and reliable.

Kubernetes is stateful. Every command can change the state of cluster and Kubernetes tries to determine needed commands to achieve that state. Launching a new pod or creating a secret ends up as a state change operation. [10]

One cluster contains at least one **master**, but preferably more to ensure high availability. Its task is to receive commands from a developer and execute them on a cluster. For example, a developer might want to launch a pod. Master receives a command and executes tasks to match with the new desired state. [10]

One cluster contains at least one **node**. It can be a VM or physical server. Nodes run tasks and applications in a cluster. A node can run limited number of pods depending on its resources and network configuration. [10]

A **pod** is the smallest unit in Kubernetes cluster. It consists of one or more containers which share allocated resources for their pod. A pod is a single instance of a given application. Scaling your application horizontally means running multiple pods. [11]

Service enables exposing as set of pods inside cluster. It works as a load balancer for them and assigns one DNS name for a set of desired pods. This helps in horizontally

scaled systems when, for example, web traffic needs to be distributed across multiple pods. [12]

An **ingress** manages access from external sources to the services in cluster. Typically, this traffic uses HTTP protocol, but can be something else too. Routes are defined by rules on ingress resource. [13]

An **ingress controller** is required to make ingress resource work. They define which external resources are needed for the ingress to expose a service. There are multiple type of ingress controllers and one can choose any of them. [14] Some cloud platforms provide their own ingress controllers. For example, in AWS you can use ALB Ingress Controller which creates an Application Load Balancer in your AWS account. [20]

By default, containers are run in stateless mode. It means that a container starts always with a clean state. All files created during its lifetime will be deleted on stop. This is where **volumes** are needed. [15]

First, volumes allow persisting data between container restarts. [16] If one is running database server in a container, it's no longer stateless. Data needs to be saved on a volume mounted to the container. Persistent volumes, even when pod is destroyed, will be covered in chapter 4.6.8.

Second, a pod can contain multiple containers. If they want to share data together, a volume needs to be created. Mounted volumes must be specified explicitly in configuration files for each container.

There are several types of volumes one can create in Kubernetes cluster. Every cloud provider, like AWS, Google and Azure, have their own volume type. [15] One special type is persistent volume and it will be covered in next chapter.

Persistent volumes solve the problem with lifecycle of normal volumes. Persistent volumes won't be destroyed when a pod ceases to exist. Persistent volumes are tightly coupled with persistent volume claims. The workflow for using persistent volumes is following:

1. User requests a persistent volume.

2. User requests a persistent volume claim, which will use the requested persistent volume.
3. Pod mounts the persistent volume to its container(s) as a volume.

Same as with volumes, persistent volumes can be one of various types. For example, in AWS you can choose to create an EBS-volume which will be mounted on the node where target pod is running. [16]

Namespaces become useful when one cluster has multiple users and, for example, they are developing many projects. Namespaces act as a scope for its resources. Different namespaces can contain resources with same names, but they must be unique within a namespace. They also provide support to limit resource usage within a namespace. [17]

Kubernetes contains a per-namespace storage for **secrets** that are needed for pods or other resources. Secrets shouldn't be included in a container image, but they should be retrieved from a storage when launching a container. For example, one could launch a database container, retrieve the root password from secrets and save it to the container's environment variables. Secrets can also be mounted as volumes when a container is launched. [18]

Deployments contain a desired state that a deployment controller in Kubernetes tries to achieve. Deployment can consist of a group of pods, replica sets, which will be deployed to the cluster. One can create two replicas of a pod at first. Later, when desired, this can be horizontally scaled by defining that the deployment must contain five replicas of the same pod. If scaling fails, then previous state will be reverted. [19]

4.7 Helm

Helm is used to manage Kubernetes applications and resources. Helm uses charts, written in YAML, to package multiple requirements and resources to run an application. For example, chart can contain a deployment, service and ingress definitions to install a web application. [21]

Charts are parameterized. It means that values are defined in separate YAML-files. This makes it easy to define different values for stage and production environments and still use the same charts. For example, for the test environment it might be enough to have two replicas of a pod running, but for the production four are needed. [22]

Helm currently requires that a component, Tiller, is installed to the cluster. It can be installed easily with a command. After that Helm commands interact with Tiller when doing changes with charts. Tiller communicates with Kubernetes API to add, modify or remove Kubernetes resources. It also keeps track of the installed charts and their versions. [23]

Charts can be installed from chart repositories or local hard drive. By default, Helm client is configured to retrieve charts from the official Kubernetes chart repository. There are many other public repositories where developers and publish and share their charts to others. [23]

5 Solution Development

This chapter describes most of the actual development work required for this study.

5.1 Requirements

The main objective of this study is to build an application, Provet Flow, which acts between Dialogflow and Provet Cloud. In the following image all components can be seen laid out with connections and communication channels.

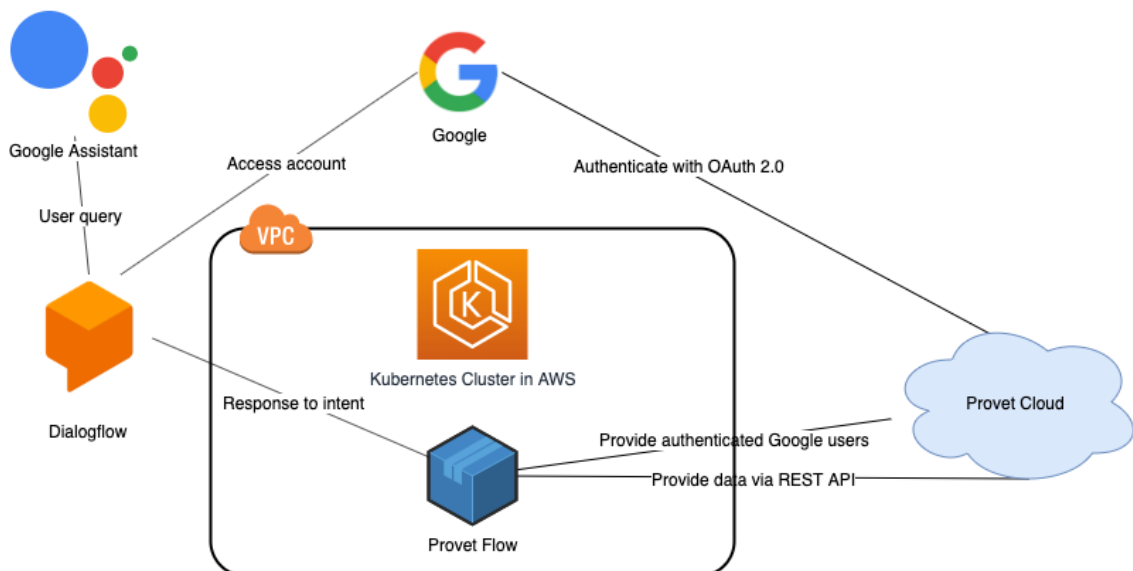


Figure 5. Architectural diagram.

5.2 Additional Development in Provet Cloud

This chapter describes additional development requirements for this study in Provet Cloud. There were two requirements to be done.

Users must be mapped to their Google account and that information must be available in Provet Flow. To accomplish, this Provet Cloud was extended with a new authentication backend Google OAuth 2.0. This feature is useful for users in general, not just for this

study. Now they can link their Provet Cloud users and authenticate with a Google account.

Information about linked Google accounts was needed in Provet Flow. Provet Cloud needed some extensions to communicate with Provet Flow. Furthermore, REST API client was built which sends information to Provet Flow about linked users and their Google accounts.

5.3 Dialogflow Agent

This chapter describes development of the Dialogflow agent.

5.3.1 Intents and Fulfillment

The Dialogflow agent was developed in Google's online platform. It is an essential part of this study as it provides speech-to-text capabilities. The agent receives speech from Google Assistant, parses and maps it to the correct intent which then makes a request to Provet Flow. Three intents were created for this study. Two intents were created to help and verify the integration with Provet Flow and the third intent was created to fulfill the proof-of-concept need. All intents are connected to the fulfillment webhook which sends all requests to Provet Flow from the Dialogflow agent.

SignIn intent initiates the sign in process and asks permission to share Google account information with Provet Flow. The user is required to provide an audible or written response "yes" two times to fully grant permissions.

Authenticate intent can be used to verify connection all the way to Provet Cloud. It tries to retrieve information about the connected Provet Cloud environment. If the integration hasn't been fully completed, the user will be provided with answer: "You are not connected to Provet Cloud". This can be the case where the user hasn't completed all the steps in Provet Cloud regarding to the Google account linkage as written in the chapter 5.4.2.

Appointments intent can be used to retrieve appointment information from Provet Cloud. For example, following audible request can be made: "What are my appointments for tomorrow?". Provet Flow will then try to find appointments for tomorrow for the connected

user. The word 'tomorrow' maps to a date so it can be a specific or a relative date. Dialogflow does automatically all date parsing which results to a ISO formatted date representation.

The screenshot displays the Dialogflow console for the 'Appointments' intent. The left sidebar contains navigation options: Intents (selected), Entities, Knowledge (beta), Fulfillment, Integrations, Training, Validation (beta), History, Analytics, Prebuilt Agents, Small Talk, Docs, Standard Free (with an Upgrade button), and Support. The main content area is titled 'Appointments' and includes a 'SAVE' button. Below the title are sections for 'Contexts', 'Events', and 'Training phrases'. The 'Training phrases' section contains three phrases: 'Add user expression', 'How does my day look on 29th September?', and 'What appointments do I have next Friday?'. The 'Action and parameters' section shows a table with columns: REQUIRED, PARAMETER NAME, ENTITY, VALUE, and IS LIST. The table contains two rows: one for 'date' with entity '@sys.date' and value '\$date', and another for 'Enter name' with entity 'Enter entity' and value 'Enter value'. A '+ New parameter' link is visible below the table.

Figure 6. Intent for appointments.

In the figure above *Appointments* intent can be seen. It contains three training phrases and detected parameter, date, has yellow background in each phrase.

As said, in this study a web application was built to handle communication between Dialogflow and Provet Cloud. Due to this webhook as fulfillment type was chosen. Each user query sends a POST request to Provet Flow with an authorization token in headers. Provet Flow checks that token is correct, parses the request, matches the user, retrieves information from Provet Cloud and finally provides an audible output to the user.

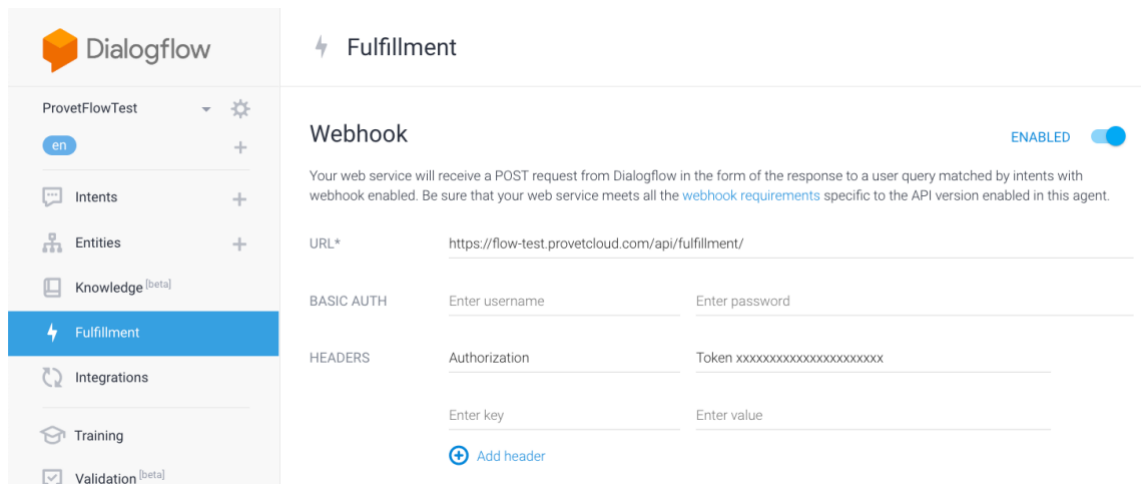


Figure 7. Fulfillment.

5.3.2 Deployment of the Action

Action is the deployment package of a Dialogflow agent, and all other things included. It wraps everything together for a proper deployment.

The deployment process of an Action is rather complicated as Google requires various verifications for the created application. This is needed to protect users from abusive Dialogflow agents that might, for example, attempt identity theft or some other malicious act.

At first company details and other general information was needed to fill it. The developer contact is always needed but, for example, marketing and business contact are optional. Other general details include description and the logo of the application. It is also important to provide privacy policy and terms of service in the details. Privacy policy is always required, but terms of service is optional in some cases. In this study it was required as the user's Google account information is needed.

The second step is to verify your brand. One can verify brand by connecting an existing website or an Android application to your agent. In this study brand verification was done by connecting the website of Provet Cloud. The process sent an email to our website administrator who then approved the verification.

After brand verification one should decide location targeting and surface capabilities. This means that one must choose countries where the application is available. Provet

Flow was decided to be available in Finland, Sweden, Estonia, United Kingdom and the United States.

Surfaces your Actions will trigger on

Answer the following questions to determine which surfaces your Actions supports. [Help](#)

Do your Actions require audio output? ⓘ Yes No
E.g., your Actions need to play a song or special tone

Do your Actions require a screen output? ⓘ Yes No
E.g., users need to view pictures or photos to interact with your Actions

Do your Actions require media playback? ⓘ Yes No
E.g., your Actions need to be able to use the MediaResponse API to play audio

Do your Actions require a web browser? ⓘ Yes No
E.g., users must be able to view content in a web browser to use your Actions

Surfaces your Actions will trigger on ⓘ










 Mobile devices (e.g. mobile phones, tablets)
 Speaker (e.g. Google Home)
 Feature phones
 Smart displays (e.g. Google Home Hub)
 Android TV
 Android Auto
 Wear OS devices (e.g. watch)
 Assistant-enabled headphones
 Phone Line with Google Assistant

Figure 8. Surface capabilities.

For surface capabilities one can choose device types where the agent can be called. For example, if the agent requires a display to show some information, then it can't be used on a Google Home speaker. For this study audible output was only required, at least for now, so it is available in most of the device types.

The last step is to release your Action. It is possible to do an alpha and beta release before actual production release. Pre-releases are not required, but they provide good insight and early feedback from a small number of users. For this study preliminary testing was done in a beta release before the production deployment. Production deployment wasn't done during this study.

5.4 Provet Flow

This chapter describes all components and features of Provet Flow.

5.4.1 Technology Stack

Application is powered by Python using Django framework. It was chosen due to its robustness and previous knowledge of the author. Also, the decision became easier as there exists a library which allows authentication to Django application with Google's OAuth 2.0.

The Django application is running inside a Docker container. Internally uWSGI is actually executing Python code with Nginx as proxy. Combination of uWSGI and Nginx is a very popular combination to run Django application. It was chosen due to simple configuration.

PostgreSQL was chosen for database. Django has a very good support for that and nowadays it's the most recommended one when working with Django. [3]

Kubernetes was chosen as platform to run the whole stack. Our company had already Kubernetes clusters, so it was an easy choice. Reserving a new server creates more costs. Our Kubernetes cluster had space still for all the needed pods. The application is designed to be highly available so two pods run the application and two pods the database.

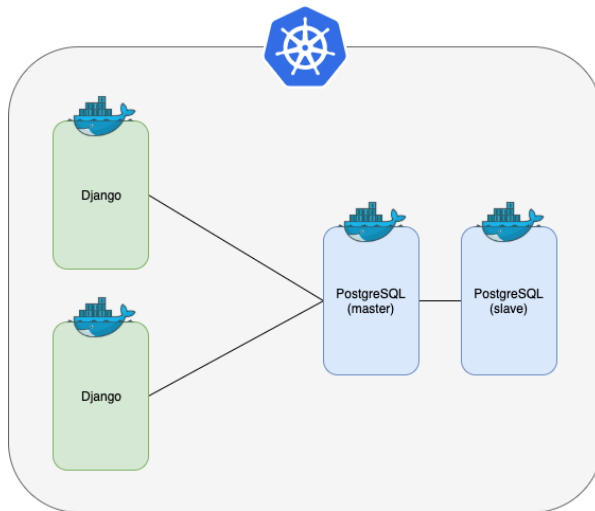


Figure 9. Technology stack.

5.4.2 User Workflow

User workflow consists of three steps as described in Figure 10: linkage, authentication and validation. It is assumed that a user has a working Google account and has access to Provet Cloud. Creating a Google account or a Provet Cloud won't be covered in this study.

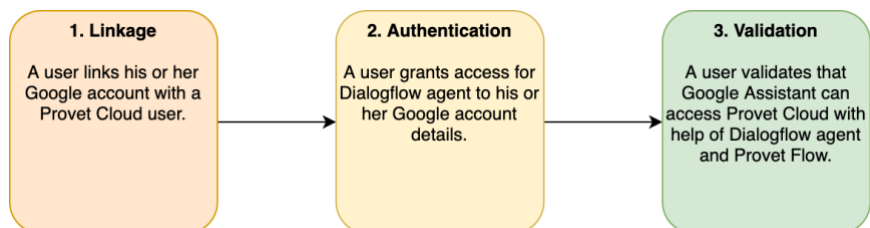


Figure 10. User workflow.

At first a user needs to link his or her Google Account with Provet Cloud user. This happens in Provet Cloud by clicking button Connect as shown in Figure 10. This opens Google authentication page (Figure 11).

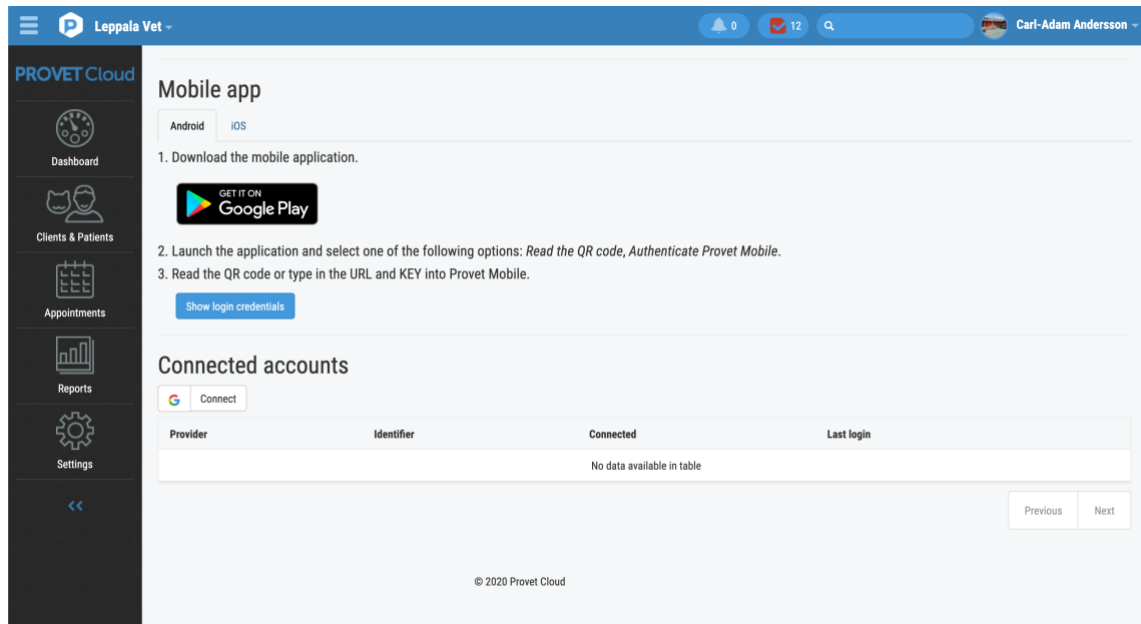


Figure 11. Connect a Google account in Provet Cloud.

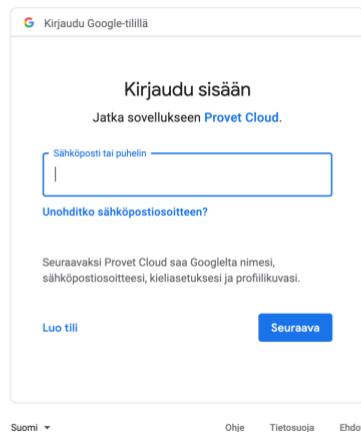


Figure 12. Google authentication.

After successful authentication Provet Cloud receives information about the connection. From here the user can continue to activate connection to Provet Flow by clicking button *Google Assistant*.

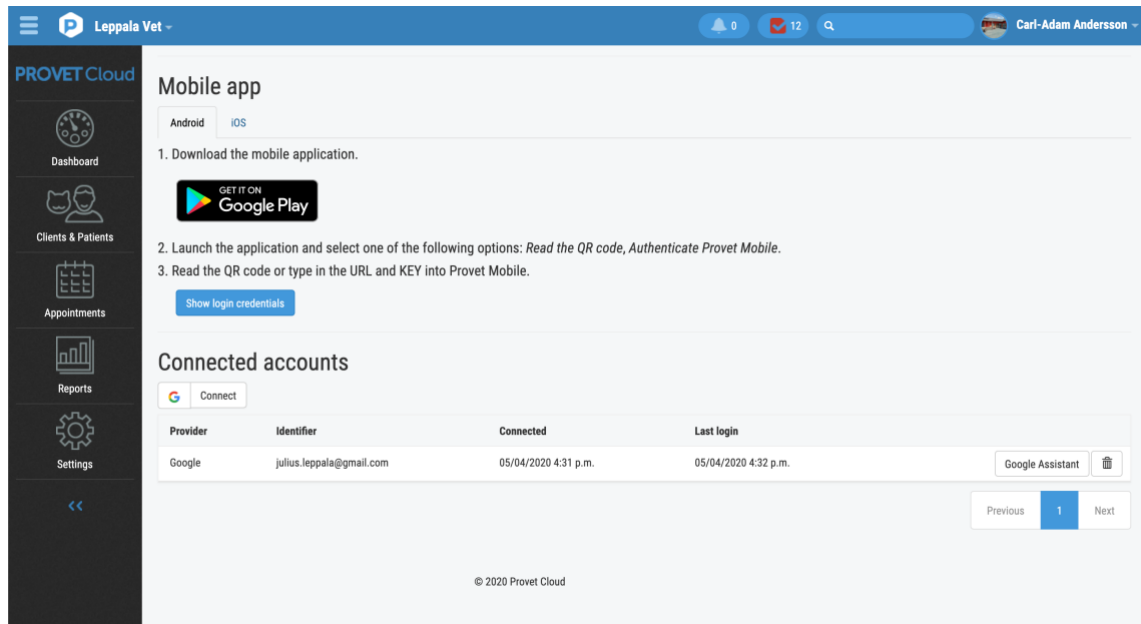


Figure 13. Successful authentication with Google.

If the user is activating Google Assistant in Provet Cloud for the first time, the following modal will be shown (Figure 14). By clicking button *Register*, information about user's Google account will be transferred to Provet Flow and a REST API token will be created. Provet Flow uses that token to fetch data in behalf of the user.

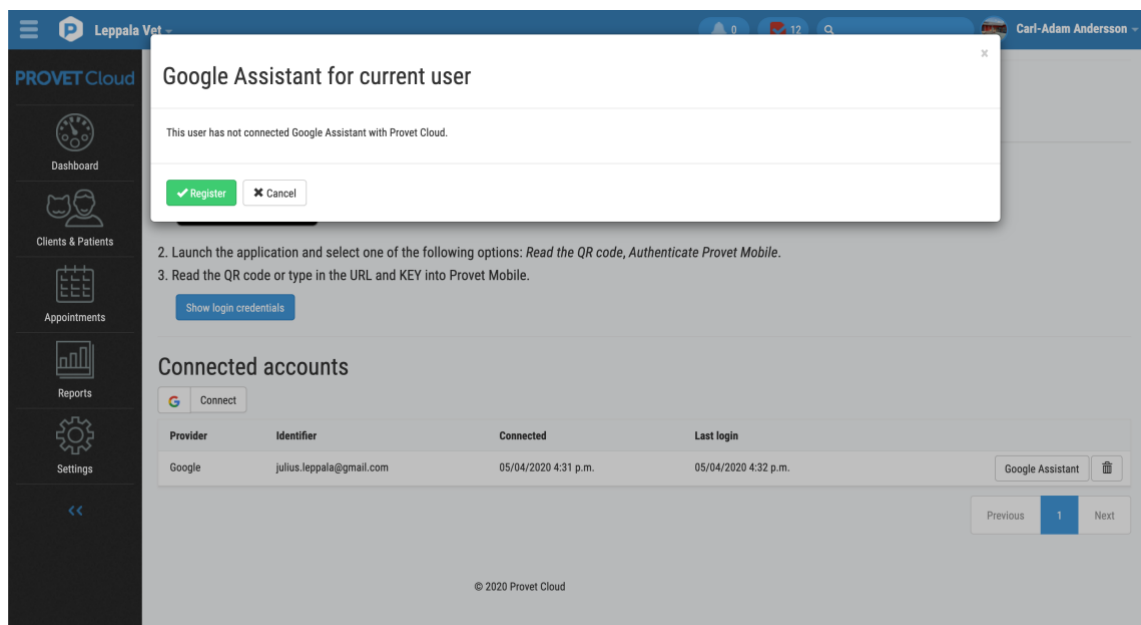


Figure 14. Registration.

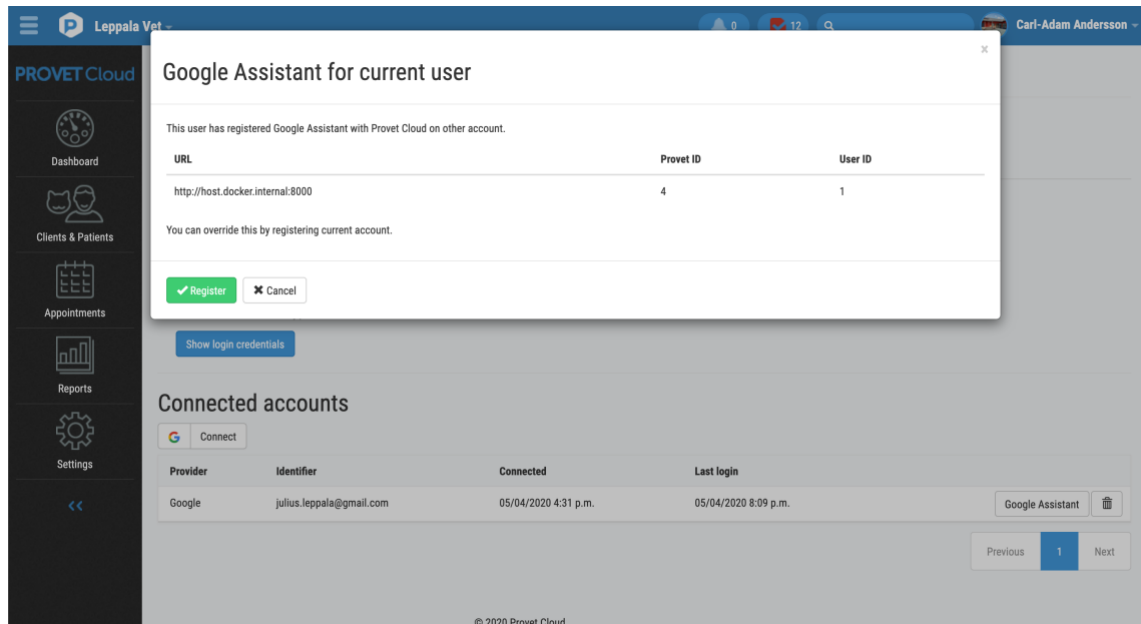


Figure 15. Details about Google Assistant connection.

After linkage a user needs to allow authentication with the Dialogflow agent. This happens by opening Google Assistant and speaking “Talk to Provet Cloud”, which activates Dialogflow agent. Google Assistant answers with a confirmation and common instructions. Then a user needs to say “I want to sign in” which will then grant access for the Dialogflow agent to user’s Google account information.

The last step is validation. It allows user to validate that all the steps have been completed and the connection is open all the way to Provet Cloud. A user can say “Authenticate” and Google Assistant answers with some details of the account in Provet Cloud.

In the user ever wants to revoke connection between Provet Cloud and Google Assistant, he or she can go to Provet Cloud where the registration was made and click the button *Google Assistant* which will open following modal.

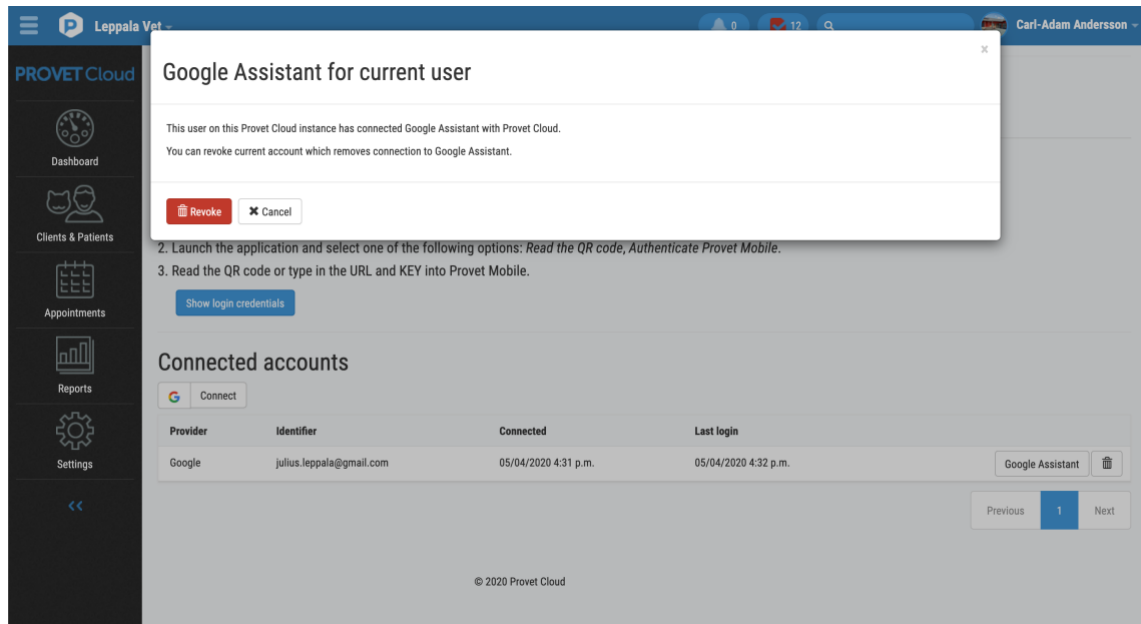


Figure 16. Revoke connection to Google Assistant.

This removes user information and REST API token from Provet Flow permanently. This can be verified by saying "Authenticate" to Google Assistant and it will reply by saying that the user is not connected to Provet Cloud.

5.4.3 GDPR

GDPR is a data protection regulation in the European Union and European Economic Area which became mandatory to adopt in May 2018. It regulates that members of the European Union are required to process and protect personal data. It also dictates that individuals must have control of their personal data and that it can be purged within certain limits. [24]

Google is a global company which has multiple data centers around the globe [25]. It means that it cannot be ensured where data goes when talking to Google Assistant and receiving answers from it. This uncertainty creates a limitation of what can be and cannot be sent from Provet Cloud to Google Assistant. Basically, it is required that a person must not be identifiable from the data that is being sent, because Google collects data from conversations with Google Assistant. [26]

REST API of Provet Cloud can return responses with personal details, because some services and third-party integrations rely on those. Provet Flow receives those responses

and does parsing to avoid exposing personal details like identity numbers or addresses, but this cannot be detected always. For example, a client has a field for identity number, and it won't be exposed via Provet Flow. However, if a user has inputted an identity number in some other field that is considered to be safe, it won't be protected anymore.

The case company takes GDPR seriously as animal health data contain personal data from owners of the animals. When creating intents each response is checked carefully and only essential data will be sent as the response to Google Assistant to avoid exposing data where a person could be identified.

5.5 Deployment Workflow

Repository

Provet Flow's source code is stored in on-premise GitLab repository and it is not publicly available or distributable. The repository is owned by the case company. Development happens in separate branches and changes are merged to the master-branch when ready.

Helm Charts

Provet Flow needs two Helm Charts to run in test and production environments. One chart is needed for database and the other for the application itself.

PostgreSQL Helm Chart is built by Bitnami. It offers production-ready charts which containers are secure and tested properly [27]. Provet Flow uses chart to run the database in test and production environments. Both are running in master-slave mode which offers high-availability for the application.

The other chart is created by the author to run Provet Flow. It describes all containers, services, health checks and other components needed to make the application run in Kubernetes. Updates are handled by Helm Charts automatically as described in 5.5.3.

CI/CD Pipeline

Provet Flow's code will be built and tested using GitLab runners. It has pipeline with three stages: Build, Test and Deploy. If tests complete successfully, the new code is deployed to Kubernetes using Helm charts.

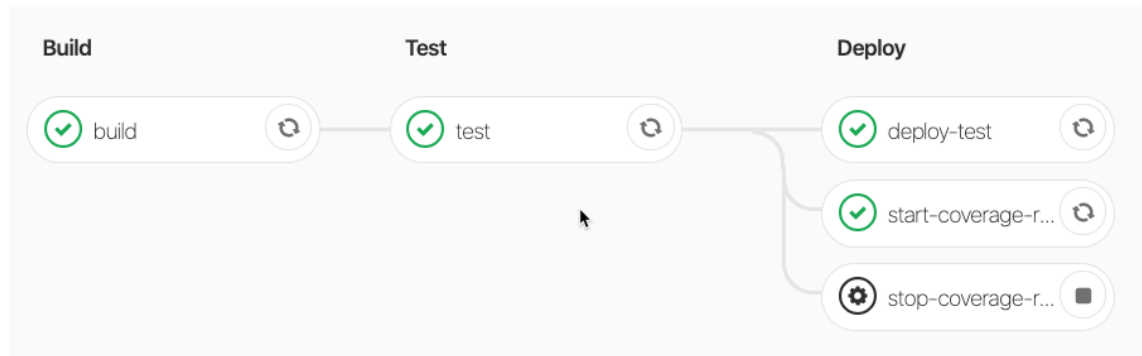


Figure 17. CI/CD pipeline.

Pipeline is triggered for all commits in any branches, but only commits to the master-branch trigger deployment.

Environments

Currently Provet Flow has only one environment. In future a separate test and production environments might be created, but for now separation is not needed.

5.6 Future Development

Future development for Provet Flow consists of three main development targets: localization, creating more intents and enabling two-way communication. Provet Cloud is an international application and used in many countries. Unfortunately, Google Assistant is not available in every country. The first priority would be to add Swedish localization to Provet Flow as Provet Cloud has many customers there.

Adding more intents is a continuous development. The case company is planning to do a survey and gather answers from users and the most needed intents that would be useful in Provet Flow. Every intent requires a reasonable amount of work as responses

need to be built always from scratch. Especially if continuing conversations will be supported then it will be even more work to be build useful and smart intents.

Enabling two-way communication means that Provet Flow would be able to execute write operations in Provet Cloud. Current proof-of-concept allows only doing read operations. Write operations are more difficult as one needs to solve at least following topics: parameter types, error handling and continuing conversations.

The last future development idea is a full release to production. Currently this proof-of-concept is in the testing phase which means that all users must be invited to test the application. Actual production release needs still to be verified by Google.

6 Results and Analysis

This chapter describes tests and results of this study. A usability test was carried out with a colleague and a person working at a veterinary clinic. In addition, a technology and code review were done with a system's architect to find any code-level vulnerabilities or issues.

6.1 Usability Test Case

This test case consists of the full user workflow as described in chapter 5.4.2. A user needs to do account linkage, registration to Provet Flow and validation in Google Assistant. After this has been completed, the user must test the intent that has been created for this study.

There are some preconditions for this test case. A user needs to have a valid Google account and he or she is willing to use it for executing the test case. Existing account in Provet Cloud is not needed as a test account has been created for this test case.

After the test had been completed comments and feedback were gathered for results and analysis.

6.1.1 Test Results From a Colleague

Testing was done remotely with a colleague, located in Finland, who is also the supervisor of this study. He was provided with instructions and also guided through the process in a remote call. The tester was very familiar with Google Assistant and his device was Google Home Mini speaker.

Generally, the tester was very impressed about the integration with Google Assistant. There were some minor issues found during the tests and those will be fixed in near future. For example, linking Google account inside Provet Cloud should more informative thus more text should be added in the modals that are shown to the user. One issue was found in *Appointments* intent as the tester came up with a different way of asking for appointments and the agent couldn't map it to the correct intent. This thing was fixed during the testing, because only one new training phrase was required for that intent.

The tester provided very good ideas for future development. The main focus was on patient related data. The Dialogflow agent should be extended that it offers a patient as context and then one can request more information about patient's medical history, for example, received medications and diagnoses. Also, it should be possible to retrieve medical history from a specific date.

The tester concluded that after these suggested additions this proof-of-concept could be presented in a business affair. Currently it could be used for demo purposes.

6.1.2 Test Results From a Veterinary Professional

Testing with a veterinary professional was done remotely, located in Sweden. She was provided with same details as the first tester. However, before the actual testing the tester was briefed about the project as this was the first time she heard about it. The tester was familiar with Google Assistant but hadn't used it much and her device was iPhone 11.

At first there were challenges to get everything working as the tester had her iPhone in Swedish language. The Dialogflow agent was only available in English. Eventually it was required that the tester changed the language on the phone itself to English as it was impossible to change the language in Google Assistant only. After these changes it was possible to start testing.

The tester was pleased about the integration with Google Assistant. However, she didn't see it being used at the clinic. Usual workday is busy, there's background noise and personnel usually don't carry smartphones with them all day long. The personnel will find the needed information quicker on clinic's computers. Instead the tester stated that this integration could be used at home if you want to know and prepare your next workday. For example, she found it useful to query your incoming appointments on the next day.

The tester gave good improvement ideas. She said that it would be useful to hear the reason for the appointment when you're asking appointments on the next day. In addition, she told that the most important information from previous consultations are diagnoses, reason of the visit, clinical notes and prescribed medicines.

6.2 Technology Review

During this thesis the author studied serverless technologies in other work-related matters. It also raised a question, whether the proof-of-concept of this study could have been built a serverless platform too. It wouldn't change results of this study, but it would affect on the costs to run the application. Possible solutions on serverless platform in AWS would've been AWS Lambda and Amazon DynamoDB.

AWS Lambda allows build individual functions which run in an isolated environment when triggered. Costs depend on resource reservation and execution time of the function. This mean during idle time, when the function is not triggered, no costs occur. However, this means that there's a slightly increased response time as all Lambda functions are triggered by demand. Execution time gets faster when called many times as it warms up. [28] [29]

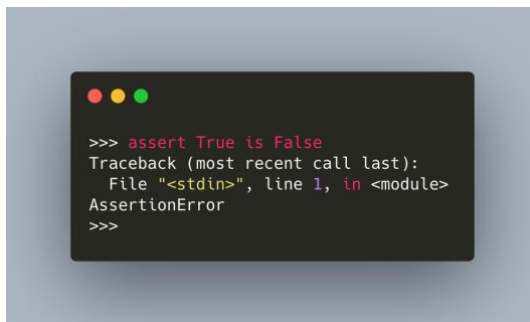
One possible serverless database is Amazon DynamoDB. It works in similar manner as AWS Lambda – you only pay for the number of requests, data storage and data transfer [30]. Provet Flow doesn't require a lot of space to save user information, so it would've been very cheap to run it on Amazon DynamoDB.

Currently it is not convenient to change technology stack. Django applications can be run on AWS Lambda, but to migrate database and code from PostgreSQL to Amazon DynamoDB can take a lot of time and cost more money. These learning points could affect on future decisions when building microservice applications in general. Depending on the estimated usage, for example, the number of incoming requests, it might be wiser to build a microservice on a serverless platform where billing is calculated based on duration needed to execute the code. In typical virtual machine approach, you have to pay for duration the virtual machine is running even when it's completely idle.

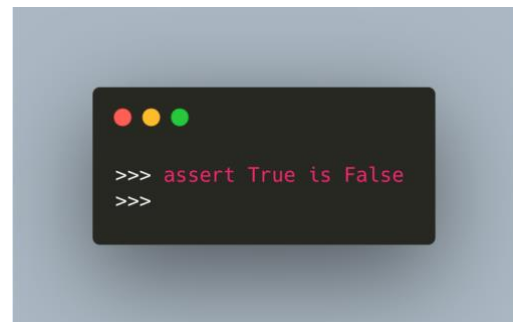
6.3 Code Review

Code review was carried out by two other developers for changes done in Provet Cloud and Provet Flow. In Provet Cloud main changes included Google account linking and a REST API client to use Provet Flow. Provet Flow code included a small Django project. Code reviews were performed in GitLab using merge requests.

Provet Cloud code review results were generally good. Some issues needed to be fixed. The most notable issue touched security in Python language and usage of assert statement. Provet Cloud uses open source library called “django-allauth” to enable authentication with a Google account. Needed components were modified to match Provet Cloud’s needs. This included reusing of some places where assert statement was used to verify states of user authentication. This was dangerous as assert statement can easily be skipped when compiling a Python programs with optimizations enabled [30].



```
>>> assert True is False
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AssertionError
>>>
```



```
>>> assert True is False
>>>
```

Figure 18. Assert statement without optimizations. Figure 19. Assert statement with optimizations.

Above there are two example executions of assert statement. The latter one is executed with environment variable PYTHONOPTIMIZE set to 1. This causes assert statement to be skipped and no AssertionError is raised. As the result of this all used assert statements were changed to normal exceptions to keep application secure.

Provet Flow’s code review was passed with minor issues. Only some unnecessary code had to be removed as it was no longer being used. Code review was performed on all code of the project.

7 Discussions and Conclusions

Virtual assistants have been around for quite a while. Apple's Siri was released in 2011 and Google released their own Google Assistant in 2016. Amazon joined the group by developing Amazon Alexa in 2014. During the last few years many applications have been released on these platforms. However, none was found to be integrated in a practice management system used in veterinary clinics or hospitals.

Google Assistant was chosen as the platform for the proof-of-concept application as the case company had some activity in Google Cloud already. The case company didn't have any existing applications using speech recognition, so this study was in completely new technical field.

The goal of this study was to integrate Google Assistant with Provet Cloud. This goal was achieved as seen in the results. Google Assistant was able to retrieve data from Provet Cloud. As a side product one can login to Provet Cloud using his or her Google account. This new authentication method will be made available to all Provet Cloud users later this year.

From the technical perspective this project was success, but still something was found that could be improved. Mainly costs of the proof-of-concept application can be high, because it is idle most of the time, at least for now. More serverless approach could be used to achieve decrease in infrastructure costs. In the other hand, the case company had Kubernetes cluster already running with some spare space, so it was convenient to place it there.

Usability tests gave good insight of the user experience. It was helpful to encounter localization issues already in the beginning as they are a very important part of the future development. Usability testers gave good development ideas and use cases. However, a general opinion was that this proof-of-concept is not usable in a veterinary clinic, but at home outside of working hours. During workdays there's a lot of distraction at the clinic which can prevent using of Google Assistant.

In conclusion this whole project was a success in its defined scope. The goal was achieved and Provet Cloud, a cloud-based practice management system, was possible to integrate with Google Assistant. However, it's highly improbable that Google Assistant

can be used at work in a veterinary clinic due to surrounding distractions. A probable use case is at home when one wants to prepare for the next workday. Some additional development is still needed and recommended before the proof-of-concept can be shown as a demo in a business affair.

References

- 1 Rouse, M., What is multi-tenancy. Web article. <<https://whatis.techtarget.com/definition/multi-tenancy>>. Accessed 12 Oct 2019.
- 2 Wiggers, K., Google Assistant will soon be on 1 billion devices, but still can't speak like John Legend. Web article. <<https://venturebeat.com/2019/01/07/google-assistant-will-soon-be-on-1-billion-devices/>>. Accessed 19 Apr 2020.
- 3 Kumparak, G., Google acquires API.AI, a company helping developers build bots that aren't awful to talk to. Web article. <<http://tcn.ch/2deWt01>>. Accessed 10 Sep 2019.
- 4 Dialogflow concepts. Web article. <<https://cloud.google.com/dialogflow/docs/concepts>>. Accessed 28 May 2020.
- 5 OAuth. Web article. <<https://en.wikipedia.org/wiki/OAuth>>. Accessed 28 May 2020.
- 6 Introducing Django. Web article. <<http://web.archive.org/web/20180729171111/https://djangobook.com/introducing-django/>>. Accessed 27 Sep 2019.
- 7 Django documentation. Web article. <<https://docs.djangoproject.com/en/2.2/ref/databases/>>. Accessed 21 Sep 2019.
- 8 dotCloud – About. Web article. <<https://web.archive.org/web/20140702231323/https://www.dotcloud.com/about.html>>. Accessed 27 Sep 2019.
- 9 McLuckie, C., From Google to the world: the Kubernetes origin story. Web article. <<https://cloud.google.com/blog/products/gcp/from-google-to-the-world-the-kubernetes-origin-story>>. Accessed 28 Sep 2019.
- 10 Concepts – Kubernetes. Web article. <<https://kubernetes.io/docs/concepts/>>. Accessed 28 Sep 2019.
- 11 Pod overview. Web article. <<https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/>>. Accessed 28 Sep 2019.
- 12 Service. Web article. <<https://kubernetes.io/docs/concepts/services-networking/service/>>. Accessed 28 Sep 2019.
- 13 Ingress. Web article. <<https://kubernetes.io/docs/concepts/services-networking/ingress/>>. Accessed 28 Sep 2019.
- 14 Ingress Controllers. Web article. <<https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/>>. Accessed 28 Sep 2019.
- 15 Volumes. Web article. <<https://kubernetes.io/docs/concepts/storage/volumes/>>. Accessed 28 Sep 2019.

- 16 Persistent Volumes. Web article.
<<https://kubernetes.io/docs/concepts/storage/persistent-volumes/>>. Accessed 29 Sep 2019.
- 17 Namespaces. Web article.
<<https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>>. Accessed 28 Sep 2019.
- 18 Secrets. Web article. <<https://kubernetes.io/docs/concepts/configuration/secret/>>. Accessed 28 Sep 2019.
- 19 Deployments. Web article.
<<https://kubernetes.io/docs/concepts/workloads/controllers/deployment/>>. Accessed 28 Sep 2019.
- 20 ALB Ingress Controller on Amazon EKS.
<<https://docs.aws.amazon.com/eks/latest/userguide/alb-ingress.html>>. Accessed 28 Sep 2019.
- 21 What is Helm. Web article. <<https://helm.sh/>>. Accessed 29 Sep 2019.
- 22 The Chart Template Developer's Guide.
<https://helm.sh/docs/chart_template_guide/>. Accessed 29 Sep 2019.
- 23 Helm Glossary. Web article. <<https://helm.sh/docs/glossary/>>. Accessed 29 Sep 2019.
- 24 Regulation (EU) 2016/679 of the European Parliament and of the Council. Web article. <<https://eur-lex.europa.eu/eli/reg/2016/679/oj>>. Accessed 1 Feb 2020.
- 25 Discover our data center locations. Web page.
<<https://www.google.com/about/datacenters/locations/>>. Accessed 1 Feb 2020.
- 26 More about data security and privacy on devices that work with Assistant. Web page. <<https://support.google.com/googlenest/answer/7072285?hl=en>>. Accessed 1 Feb 2020.
- 27 Bitnami PostgreSQL Helm Chart. Web page.
<<https://bitnami.com/stack/postgresql/helm>>. Accessed 2 Feb 2020.
- 28 What is AWS Lambda. Web page.
<<https://docs.aws.amazon.com/lambda/latest/dg/welcome.html>>. Accessed 13 Apr 2020.
- 29 How to keep your Lambda functions Warm. Web page.
<<https://read.acloud.guru/how-to-keep-your-lambda-functions-warm-9d7e1aa6e2f0>>. Accessed 13 Apr 2020.
- 30 Amazon DynamoDB pricing. Web page.
<<https://aws.amazon.com/dynamodb/pricing/on-demand/>>. Accessed 13 Apr 2020.

- 31 Running python with optimizations makes UsernamePasswordMako accept any password for any user. Web page.
<<https://github.com/IdentityPython/pysaml2/issues/451#issue-256426229>>.
Accessed 26 May 2020.

Instructions to activate Google Assistant integration with Provet Cloud



Instructions to activate Google Assistant integration with Provet Cloud

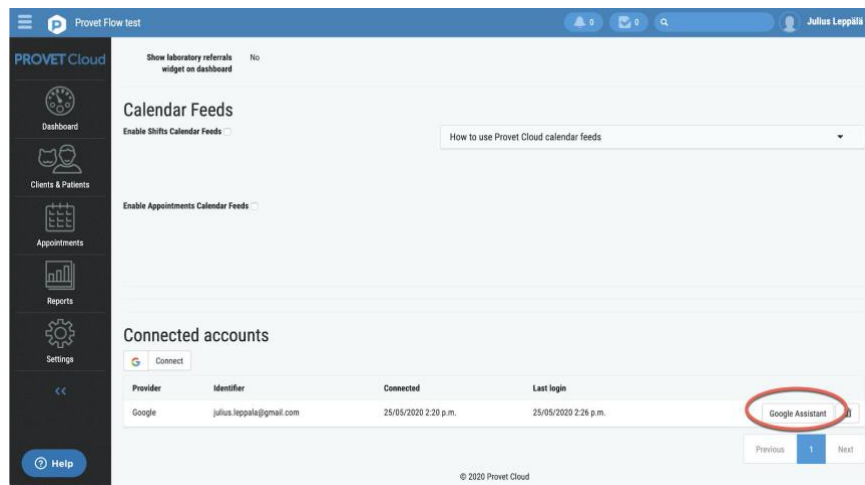
1. Link your Google account

In Provet Cloud go to your user profile and link your Google account with your Provet user.

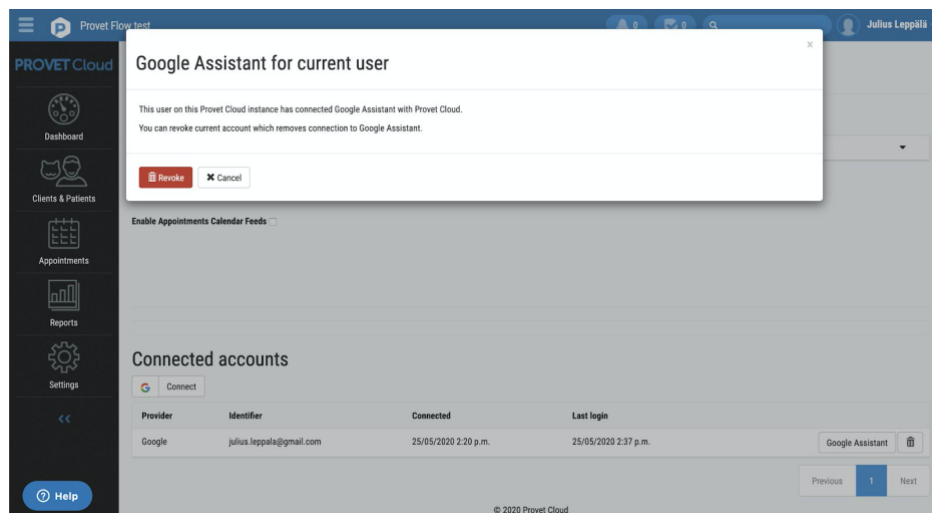
The image shows two screenshots of the Provet Cloud dashboard. The top screenshot shows the user profile menu, with the 'User Profile' option circled in red. The bottom screenshot shows the 'Connected accounts' section, with the 'Connect' button circled in red.

Connected accounts table:

Provider	Identifier	Connected	Last login
Google	julius.leppala@gmail.com	25/05/2020 2:20 p.m.	25/05/2020 2:25 p.m.



After these steps are completed you can verify connection by clicking the “Google Assistant” button again. A following modal should appear.





2. Activate Google Assistant

You can use Google Assistant with a smartphone (iOS, Android) or with another Google Assistant supported device, like Google Home speaker. Please note that **only English** language is supported.

Start by saying:

Talk to Provet Cloud

You should get a reply that says "Welcome to Provet Cloud!".

The next step needs to be done only once. You need to allow Provet Cloud to use your Google account information with Google Assistant. Next you can say:

I want to sign in

Google Assistant will confirm you about giving permissions. You will need to reply "Yes" twice in two separate questions. At the end of sign in process you will get a confirmation that you're connected to Provet Cloud if everything went well.

3. Start using Google Assistant with Provet Cloud

Now you're ready to go. Start by asking about your appointments on a specific date, for example:

Do I have appointments on 26th of May?