Installation and Dev-Ops guide
Cognigy.AI v4.X

Cognigy GmbH

March 9, 2022

Document history

Editor	Date	Version	Comment
Benjamin Mayr	02.10.2020	Revision 1	Initial release for v4.0.0
Benjamin Mayr	14.10.2020	Revision 2	Improve infrastructure diagrams; add upgrade notice; fix permissions when creating local storage
Benjamin Mayr	16.11.2020	Revision 3	Add documentation for custom theming.
Benjamin Mayr	21.12.2020	Revision 4	Release for v4.1.0. Includes additional documentation for Cognigy Live Agent.
Benjamin Mayr	08.02.2021	Revision 5	Updated documentation for custom theming.
Benjamin Mayr	11.06.2021	Revision 6	Updated documentation for CORS configuration.
Alexander Kirilyuk	10.07.2021	Revision 7	Add documentation for Management UI.
Dat Tang	09.02.2022	Revision 8	Update single-node setup recommendation
Dat Tang	24.02.2022	Revision 9	Specify supported Kubernetes versions
Dat Tang	09.03.2022	Revision 10	Remove single-node setup

Contents

1 Preamble **5**

2 Introduction **6**
2.1 Prerequisites . . . 6
2.2 Hardware requirements . . . 7
2.3 Sizing your Cognigy.AI installation . . . 8

3 Installation **11**
3.1 Cognigy.AI manifest files . . . 11
3.1.1 Templates and kustomization . . . 13
3.1.2 Using templates . . . 14
3.2 Configuring container registry access . . . 16
3.3 Secrets . . . 16
3.4 Storage . . . 17
3.4.1 Introduction . . . 17
3.4.2 Storage for multi server installations . . . 18
3.5 Database, Message-Broker and Cache . . . 18
3.6 Installing Cognigy.AI . . . 19
3.7 Retrieve login credentials . . . 22
3.8 Activating your license key . . . 23

4 Configuration **24**
4.1 Tweaks . . . 24
4.2 Custom theme . . . 30
4.2.1 Overview . . . 30
4.2.2 Asset requirements . . . 31
4.2.3 How to build a theme . . . 32
4.2.4 Configuring a theme . . . 34
4.3 CORS configuration . . . 35

5 Updating **36**
5.1 Preparation . . . 36
5.2 Retrieving the update . . . 36

6 Cognigy Live Agent **37**
6.1 Prerequisites & Hardware requirements 37
6.2 Installation 38
6.2.1 Introduction 38
6.2.2 Using templates 39
6.2.3 Secrets 39
6.2.4 Storage 39
6.2.5 Patch files 40

7 Cognigy Management UI **42**
7.1 Prerequisites & Hardware requirements 42
7.2 Installation 43
7.2.1 Introduction 43
7.2.2 Using templates 43
7.2.3 Secrets 44
7.2.4 Patch files 44

Listings

3.1 Cloning the Kubernetes repository 12
4.1 Environment variables to configure allowed CORS origins 35

Chapter 1

Preamble

This document is the official installation and operations document for Cognigy.AI. It is meant for system administrators and ops engineers who are responsible for setting up Cognigy.AI installations and who maintain the operational stability of our software product. It also provides documentation on how to monitor an installation, detect anomalies and ensure that the product delivers high performance and stability even under high load scenarios.

Cognigy.AI is a highly scalable software product and based on a modern microservices-based software architecture. A product using such an architecture consists of multiple small binaries that all have to run in order for the whole product be actually available and fully functional. The advantage of an approach like this is, that multiple processes can be spread amongst multiple servers on which these binaries get executed. So we can more easily use the power of multiple machines - and we can exchange parts of the application without stopping the whole application. The result is a product which can be scaled quite easily by adding additional hardware.

Our product is fully containerized and leverages the power of technologies like Docker and Kubernetes. We suggest that you get familiar with tools like Kubernetes as it is the container orchestrator that will make sure that all of these small services start, run and restart properly. Kubernetes also makes sure that you can update our software-platform by offering a so-called *rolling update mechanism* in which it replaces the binaries that run - one by one - in order to guarantee that the product is still available. With Cognigy.AI version 4.0.0 we have **dropped support for Docker Swarm** as a second container orchestrator. The software support for Swarm is quite weak and we can see other software vendors in the industry to drop support for Swarm as well.

> Cognigy.AI 4.X introduces a lot of breaking changes and is not compatible with Cognigy.AI 3.X. You can't upgrade your already existing installations with Cognigy.AI 4.X by updating your infrastructure. Instead, you have to setup a new dedicated Cognigy.AI 4.X environment and migrate your content.

Chapter 2

Introduction

2.1 Prerequisites

In order to set up a Cognigy.AI installation, check the following list of requirements before you get started:

- **Access to infrastructure / Cloud provider**: Installing our product requires a fully functional Kubernetes-cluster. There are many ways on how you can set up a Kubernetes-cluster. We suggest that you use one of the managed services (Amazon EKS, Microsoft AKS, Google GKE) in case you are not familiar with Kubernetes as these services offer state-of-the-art configurations and guarantee a high level of stability. Please consult the official Kubernetes documentation[1] in order to get more information on the various ways of how a cluster can be provisioned. We will not cover this topic in this manual.

  > Cognigy.ai runs on Kubernetes version between 1.19 and 1.21

- **License**: Our product requires a license key which you will receive once you have signed a license agreement with us. The key is necessary to fully install the product.
- **Container Image registry credentials**: Our product is a fully containerized software product. We are shipping our binaries in the form of Docker Images[2] which get stored in so-called container registries. Cognigy operates their own container registry from which customers can pull our images. You will receive credentials which are required in order to authenticate and pull those images.
- **Domains / DNS names**: Our product exposes several web-service for which you need to assign DNS names for. Please ensure that you have access to services which are required to create additional DNS names (e.g. a platform to manage DNS

[1] https://kubernetes.io/docs/setup/
[2] https://kubernetes.io/docs/concepts/containers/images/

entries for your domain(s) like Go-Daddy, United-Domains etc.). If our product will only be usable from within a private network, ensure that you can create DNS entries for this private network.

- **TLS certificate(s)**: In order to encrypt web-traffic in transit when using the web-services of our product, you will need to have an SSL certificate which protects the DNS names you choose for your services. We highly suggest the usage of a wildcard certificate as it makes the whole configuration process way more easy. If you can't use a wildcard certificate, make sure that your standalone certificate protects all DNS names you choose. The reverse-proxy we ship with the product will not be able to use multiple certificates out of the box. Additional configuration will be necessary if you want to use multiple certificates.

- **Whitelisting of Domains**: If you setup your environment in your own data-center, please make sure that you can reach **downloads.cognigy.com:443** as well as **docker.cognigy.com:5000** as they are required during the installation. The later is our container registry, the first one is a server from which the application will download data during the installation.

- **Network**: If you are planning to install our product in your own data-center, chances are high that there will be a firewall protecting your office network from unwanted access. Please make sure that all valid HTTP methods are allowed (GET, POST, DELETE, etc.) and can pass through your firewall. Please also ensure that you **can deal with Websockets** as the product heavily relies on Websockets. Some proxies might require additional configuration for Websockets to work properly.

> While an installation on Windows-based servers might be possible, we are not supporting Windows as an operating system nor do we test our product based on Windows-based servers. We will not be able to provide any support for our product if you pick Windows as your operating system.

2.2 Hardware requirements

Our product Cognigy.AI requires powerful hardware for its natural language capabilities and its enterprise-grade features. You will need to provision a *Kubernetes cluster* in which the applications microservices can get scheduled.

Cognigy.AIs natural language understanding capabilities support a wide variety of different languages. Customers can decide which languages they need and only deploy the necessary microservices into their clusters in order to safe hardware and related costs. This manual **contains the minimal hardware requirements** which are necessary to run our software. We provide an *Excel sheet* which can be used in order to calculate detailed hardware requirements as well as rough pricing based on your desired

configuration for larger production scenarios.

For a smaller product installation with natural language capabilities in *English language only* you will need the following hardware available in your Kubernetes cluster:

Services	Resource limits
Product dependencies	5.0 CPU, 9.0 GB
Main product services	5.8 CPU, 4.7 GB
English NLU services	2.0 CPU, 5.0 GB
Total	12.8 CPU, 18.7 GB

We also have to mention that the Kubernetes system services will also need a certain CPU and memory budget - hence it makes sense to reserve some more hardware also for future use-cases and the option to scale some of our microservices. We would, therefore, advice to have at least the following hardware available:

- **16 CPU cores, select compute-optimized machines**
- **32 GB of memory**
- **50GB + 10 GB + 10 GB of SSD storage for database and application storage**
- **128 GB of SSD storage per Kubernetes node (for Docker Images)**

> Picking general-purpose machines instead of compute-optimized machines can have a negative impact on the performance of our software. We advise you to pick compute optimized machines with high-clock speeds. This is especially important if you plan to have support for multiple natural language understanding services.

2.3 Sizing your Cognigy.AI installation

> Using testing Kubernetes tools like **microk8s** or **minikube** are not supported by Cognigy, so we cannot support if some features of the product does not work properly.

The system components of Kubernetes will run on at least 3 dedicated small servers in order to guarantee, that our products microservices do not interfer with them. There are many ways on how you can actually set up a multi machine Kubernetes cluster, some of these are:

- you can use **Amazon EKS (Elastic Kubernetes Service)** if you are using Amazon Web Services (AWS)

- you can use **Azure AKS (Azure Kubernetes Service)** if you are using Microsoft Azure
- you can use **Google GKE (Google Kubernetes Engine)** if you are using Google Cloud
- you can use **Openshift** if you are running your workloads on-premise
- you can setup your own Kubernetes cluster and provision the cluster with software like **Docker Enterprise**

> No matter which type of setup you select, installing our product depedencies as well as our actual product will only slightly differ as Kubernetes acts as an abstraction layer between containerized software products and the actual physical hardware.

In an ideal world, your Kubernetes cluster should look like in figure 2.1 - distributed across three availability zones and using an elastic load balancer. For such a loadbalancer, a fully managed product from AmazonWebServices, MicrosoftAzure or Google can be used. If you want to run the product on-premise, you might have other solutions available.

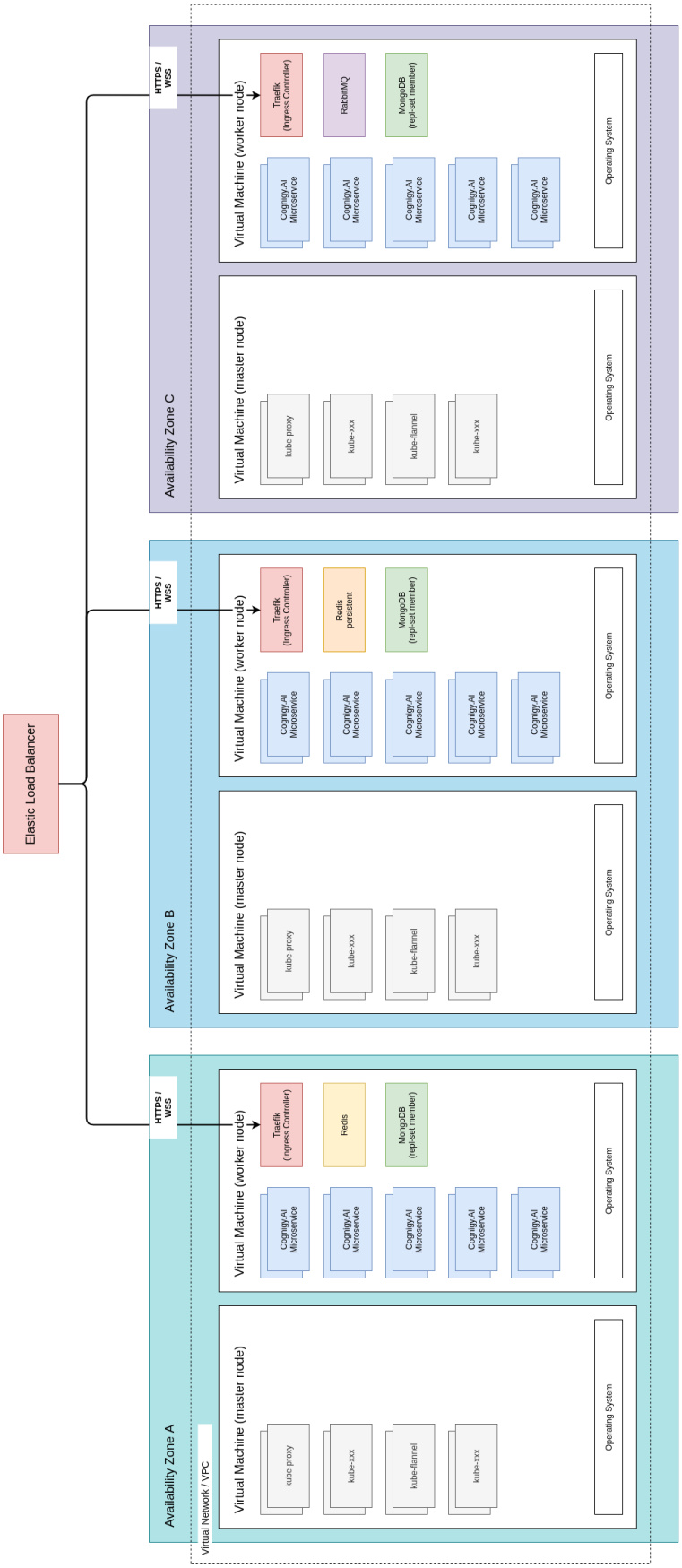

Figure 2.1: A multi-server Kubernetes cluster that spans across multiple availability zones and uses master and worker nodes for separation of duties.

Chapter 3

Installation

Installing Cognigy.AI requires a fully working Kubernetes cluster, so please make sure that you carefully read 2 Introduction in which we explain the different possibilities you have in order to provision such a cluster. Installing our product requires **full permissions** on the Kubernetes cluster you want to use, as we have to deploy multiple objects into the cluster; these include:

- Deployments
- Ingress objects
- Services
- PersistentVolumes & PersistentVolumeClaims
- ConfigMaps
- Secrets

3.1 Cognigy.AI manifest files

In order to deploy an application into a Kubernetes cluster so-called manifest files are required. These manifest files describe the application in an abstract way. They define attributes like environment-variables or storage.

Cognigy stores manifest files for Cognigy.AI in a public Git[1] repository on GitHub. In order to install our product, our repository needs to be cloned. We highly recommend to use Git and to clone the repository vs. downloading the files as a zip-archive. Working with Git will help you to update your Cognigy.AI installation in the future.

[1] Git is a decentralized version control system for text-based files.

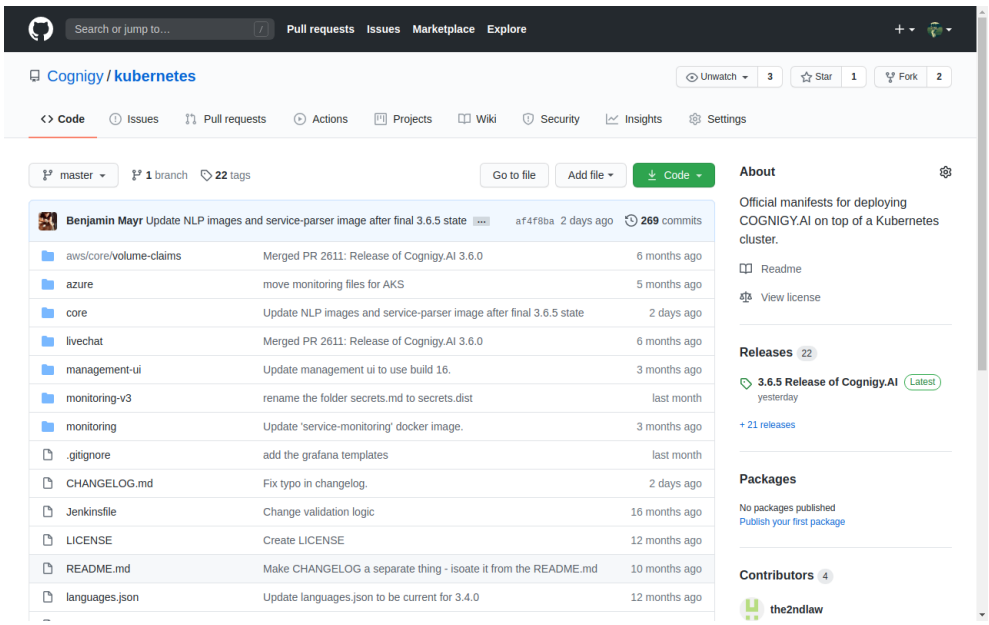

Figure 3.1: Manifest files for Cognigy.AI are stored in a public Git repository

In order to get started with the setup, please clone our Git repository. 3.1 Cloning the Kubernetes repository shows the command which you can use in order to accomplish that.

Listing 3.1: Cloning the Kubernetes repository

```
git clone https://github.com/Cognigy/kubernetes.git
```

Contents of the repository

- **cloudproviders**
  Contains files you can use in order to deploy the application into cloud environments. These files only act as examples and might not be fully complete.

- **core**
  Contains the manifest files which are necessary to deploy our core product Cognigy.AI as well as its dependencies.

- **live-agent**
  Contains an additional product called the **Cognigy Live Agent** which was introduced in Cognigy.Ai v4.1.0. Please have a look at 6 Cognigy Live Agent for additional documentation.

- **management-ui**
  Contains an additional product called the **management ui**. This product actually started as an internal tool we have created in order to manage our own

SaaS installations. Meanwhile, customers can also use it to manage their installations, create new organisations[2] and e.g. publish system messages for maintenance reasons.

- **monitoring**
  Contains additional tools you can use in order to deploy a fully compatible and pre-configured monitoring suite.
- **CHANGELOG.md**
  A file which briefly describes the changes we have made in files within this repository. The changelog contains an entry per version of our product. We will mention structural changes to files.
- **COMMANDS.md**
  Contains all commands you might need during installations. This manual heavily references commands in this file so that you can copy/paste them easily.
- **LICENSE**
  Contains the license for our manifest files. The license only relates to the manifest files, not our actual product.
- **README.md**
  Readme with brief information about the repository.

> We suggest that you read our manual side-by-side while having a look at the files in our kubernetes Git repository. Please have a look at the COMMANDS.md file as it will contain all commands you will need to use during the installation. You can often copy/paste these commands with only sligh modifications.

3.1.1 Templates and kustomization

We have completely restructured our kubernetes Git repository for Cognigy.AI version 4.0.0 as we wanted to make it easier for our customers to work with our manifest files. We are introducing a functionality called *kustomize*. Kustomize[3] allows our customers to use our manifest files and apply their own patches on top of them. By leveraging this mechanism, there is a strict separation between our files which will change with each release and your files which contain your individual configuration. This way, you can always pull our repository for lates changes while keeping your patches stable. We highly recommend reading the documentation of kustomize. Kustomize is part of *kubectl* - the Kubernetes command line interface - so you don't have to install an additional tool.

Let's further analyze the contents of the *core* folder of our repository:

[2] An organisation is a fully isolated space in a Cognigy.AI installations. Other software vendors often also refer to something like this as tenant.

[3] https://kustomize.io/

- **manifests**
  Contains our actual manifest files which are necessary to deploy our application. These files (or at least parts of it) will change with each release of our product. Please leave these files untouched - any updates to these files will be overwritten as soon as you pull the latest changes.

- **template.dist**
  Contains a blueprint of patch-files in a kustomize-compatible format. We have structured the patch files in ones that are related to our actual product (see product folder) and files you will need in order to patch our dependencies (see dependencies folder). Please also leave these files untouched. We will make a copy of this directory in which you can then start to apply your changes.

- **make_environment.sh**:
  A script which allows you to make a copy of our *template.dist* folder as well as update a copy of the *manifests* folder within your template-copy. We will use this script in the next section of this manual in order to create and prepare files for our first Cognigy.AI installation.

> Customers can use the make_environment.sh script in order to create multilple copies for multiple product installations. You might want to setup multiple Cognigy.AI installations for e.g. development, staging and production.

3.1.2 Using templates

Let us now leverage the power of the templating approach in order to prepare our own files for the first installation of Cognigy.AI. The COMMANDS.md file contains a cheatsheet with all commands that will be used during the installation. This document will always refer to this cheatsheet and individual commands so you can copy/paste them.

Open a terminal, navigate to your local copy of our kubernetes repository, make the *make_environment.sh* script executable and create a folder for your Cognigy.AI environment by executing the script with a name for your environment - we will pick **development** for this example.

> Cheatsheet, 3.1.2 Using templates, Preparing the files for a Cognigy.AI environment

After you ran the *make_environment.sh* script with the suffix for your environment, you should be able to see a new folder containing the file-structure we will use for our further work. The contents in this directory will remain stable even if you pull the latest

changes from GitHub[4]. If you inspect your new directory, you will see the following contents:

- **dependencies**
  Directory with files which we will use to deploy our dependencies (databases etc.) into your Kubernetes cluster. Sub-directories are:

  - **manifests**
    Copy of our manifests folder - don't touch these files. **You can make this copy current by re-running the make_environment.sh script with the same environment name again.**
  - **overlays**
    Contains patch files you can modify to overwrite files located in the manifests folder.
  - **kustomization.yaml**
    Defines instructions for kustomize. You can also modify this file.

- **product**
  Directory with files which we will use to deploy Cognigy.AIs microservices into your Kubernetes cluster. Sub-directories are:

  - **manifests**
    Copy of our manifests folder - don't touch these files. **You can make this copy current by re-running the make_environment.sh script with the same environment name again.**
  - **overlays**
    Contains patch files you can modify to overwrite files located in the manifests folder.
  - **secrets.dist**
    Contains blueprints of Kubernetes secrets. We offer a tool which can help you to generate random data and create a ready-made copy of these secrets for your convenience. This initialization tool will create a copy of the contents and place them into a new directory called *secrets*.
  - **kustomization.yaml**
    Defines instructions for kustomize. You can also modify this file.
  - **replica_count.yaml**
    Gives you the ability to overwrite the replica-count per microservice. This is especially important for production installations. You can also modify this file.

[4]Except the manifests folder in your environment - this is a 1:1 copy of our manifests

3.2 Configuring container registry access

Before you can go ahead, start to adjust our template-files and deploy the first manifest files into your Kubernetes cluster, we have to make sure that your cluster is able to authenticate with our container image registry in order to pull our Docker Images. You have to create a so-called *image pull secret*[5] in your cluster. Please go ahead and run the command from our cheatsheet.

> Cheatsheet, 3.2 Configuring container registry access, Creating the image pull secret

3.3 Secrets

Kubernetes offers a concept of how sensitive information can be stored in the cluster and how such information can be made accessible to applications running in clusters. A secret[6] is a peace of information that will get created once and stored securely in the cluster. Secrets are typically used for passphrases, tokens, certificates and private-keys. In 3.2 Configuring container registry access we have already created a special type of secret which Kubernetes will use internally to pull Docker images from our container image registry. Our application and its microservices use secrets in order to get access to database credentials and such.

Our product uses multiple secrets and we provide blueprints for these in the **product/secrets.dist** folder of your environment directory. Generating long and secure tokens is important as they will ensure higher security of your data. In order to simplify the process of generating random and secure secrets, we have created a small helper tool you can leverage for this job. Just navigate into your environment folder (e.g. development in our example), download the so-called **initialization** binary, make it executable and run it with the generate-flag as described in the cheatsheet. We, meanwhile, offer the initialization binary for Linux, MacOSX and Windows in order to help our customers you want to manage their Kubernetes cluster from operating systems other than Linux. Please ensure that you adjust the commands referenced in the **COMMANDS.md** file accordingly in case you are using MacOSX or Windows.

> Cheatsheet, 3.3 Secrets, Creating random secrets for Kubernetes

Running the initialization tool will create a copy of the **secrets.dist** folder (name secrets) and will fill-in random data. Running the initialization tool a second time will not have any affect as we don't want to overwrite your secrets.

[5] https://kubernetes.io/docs/tasks/configure-pod-container/pull-image-private-registry/
[6] https://kubernetes.io/docs/concepts/configuration/secret/

> Please make sure that you keep your secrets safe. Never share them and ensure that you have a backup in place. If you loose your secrets and you have to rebuild your Kubernetes cluster, the data stored in the main application database is unusable as the secrets store database credentials as well.

You can deploy your secrets (the ones in the secrets folder, not secrets.dist!) into your cluster as they will be required in order to all microservices can start. Please run the command in the cheatsheet in order to apply the manifest files.

> Cheatsheet, 3.3 Secrets, Creating secrets in your Kubernetes cluster

3.4 Storage

3.4.1 Introduction

Our product needs to be able to store data in a reliable and persistent way. Such data might be the resources of your bots, your projects, the users who can login into the product as well as the conversation history of our users. Kubernetes offers an abstract way of how our application can *see* and consume storage - it essentially works in layers like this (from bottom to top):

- **Volume mount**:
  A Pod can use a volume mount to get access to storage by mentioning a persistent volume claim.
- **Persistent Volume Claim**:
  A persistent volume claim is another object in Kubernetes which can be used to claim some parts of a persistent volume.
- **Persistent Volume**[7]
  A persistent volume is a Kubernetes object that offers a more abstract view on the raw storage.
- **Storage Medium**:
  Actual storage medium like a disk. This can be further virtualized by things like NFS.

To start building up our storage layer, we have to first select the *Storage Medium*. If you are setting up a **multi server installation** e.g. in a cloud-environment, you can utilize the files located in *core/cloudproviders*. In any case you have to make sure that you first setup the PVs[8] and the PVCs[9] before you can further proceed.

[7] https://kubernetes.io/docs/concepts/storage/persistent-volumes/
[8] Persistent Volumes
[9] Persistent Volume Claims

Please select a main storage location on your server where enough free disk space is present. We advice you to have at least the following capacity available:

- **50 GB for mongodb**
- **10 GB for redis-persistent**
- **10 GB for flow-modules/functions**

3.4.2 Storage for multi server installations

> Cheatsheet, 3.4 Storage for public clouds

3.5 Database, Message-Broker and Cache

Once the secrets were applied to the cluster, we can focus on installing the dependencies of Cognigy.AI. Our product is based on the following other software products:

- **MongoDB**
  MongoDB[10] is a popular NoSQL database and the main storage backend of a Cognigy.AI platform[11].
- **Redis**
  Redis[12] is an extremely fast in-memory key-value store. A Cognigy.AI installation depends on two separate installations of Redis - one persists onto disk and the other one only keeps the data-set in-memory.
- **RabbitMQ**
  RabbitMQ[13] is a highly performant message broker Cognigy.AI is using for inter-microservice connectivity.

You can install these products on your own and even outside of your Kubernetes cluster. This manual is not covering that part. For simplicity, we have prepare all necessary files in our kubernetes repository so you can simply deploy them into your cluster as well.

[10] MongoDB, a fast and reliable NoSQL database - https://www.mongodb.com/

[11] Please note that this database can't be replaced with other databases - our product does not e.g. offer support for Microsoft SQL Server or any other databases.

[12] Redis, an extremely fast in-memory key-value store - https://redis.io/

[13] https://www.rabbitmq.com/

The ready-made files for deploying MongoDB, Redis and RabbitMQ might need to be modified at some point. The resource-constraints we have configured for these products might be enough of you get started with using our product, but your data-set will grow over time and the values might no longer fit. You can use kustomize and the patching approach in order to patch the necessary values. Please monitor especially these depencies as they will have a massive impact on the performance an stability of our product.

Cheatsheet, 3.5 Database, Deploying our products dependencies

Once your MongoDB database-server is up, we have to connect to it through the MongoDB command line interface as we have to initialize a so-called replica-set[14]. This is necessary as Cognigy.AI 4.X heavily depends on features which do not work on standalone MongoDB deployments[15]. Connect to your database, initialize the replica-set and create all databases including users. Our cheatsheet contains the commands you can use for that. Once you have executed all commands within the MongoDB Pod, please exit from the container.

Cheatsheet, 3.5 Database, Initializing a replica-set and creating databases with users

3.6 Installing Cognigy.AI

At this point in time you should now have a fully working Kubernetes cluster with a proper storage backend and our product dependencies. You can use the commands in our cheatsheet in order to double-check that your *Persistent Volumes*, your *Persistent Volume Claims* as well as the *Deployments* for our dependencies are up as expected.

Cheatsheet, 3.6 Installing Cognigy.AI, Checking the state of your cluster

Running these commands should report Persistent Volumes and Persistent Volume Claims to be in a *bound* state. Our products dependencies and their deployments should report that you have 1/1 replicas per product dependency (MongoDB, Redis, Redis-Persistent and RabbitMQ). We now have to patch some more files in the *product* folder for your environment - we will now cover all of them.

Main ConfigMap Please open the main ConfigMap patch located under

[14] https://docs.mongodb.com/manual/replication/

[15] A single mongod process can also be configured to run in replica-set mode.

core/development/product/overlays/config-maps

and patch the DNS names that are mentioned. We have placeholders in there and you have to replace them with the DNS names you have choosen for your installation. 3.2 Adjusting the main ConfigMap patch for your DNS names shows an example of how this looks like for our SaaS v4 environment.

```
18  # DNS names
19  - op: add
20    path: /data/FRONTEND_BASE_URL_WITH_PROTOCOL
21    value: "https://app.cognigy.ai"
22  - op: add
23    path: /data/BACKEND_BASE_URL_WITH_PROTOCOL
24    value: "https://api-app.cognigy.ai"
25  - op: add
26    path: /data/ENDPOINT_BASE_URL_WITH_PROTOCOL
27    value: "https://endpoint-app.cognigy.ai"
28  - op: add
29    path: /data/WEBCHAT_BASE_URL_WITH_PROTOCOL
30    value: "https://webchat-app.cognigy.ai"
31  - op: add
32    path: /data/ODATA_HOSTNAME
33    value: "odata-app.cognigy.ai"
34  - op: add
35    path: /data/ODATA_PROTOCOL
36    value: "https"
37
38  # INTERNAL API for management ui
```

Figure 3.2: Adjusting the main ConfigMap patch for your DNS names

Ingress objects You also have to patch the Ingress objects our Ingress controller[16] uses in order to route incoming requests to the correct microservices. The patch-files that need to get modified are located under

core/development/product/overlays/reverse-proxy/ingress

You have to patch all five files - service-analytics-odata_patch, service-api_patch, service-endpoint_patch, service-ui_patch and service-webchat_patch. Please just replace our examples and ensure that you are not adding any protocoll (http or https) here as they should only contain the DNS names. A valid DNS name for the API ingress route patch would e.g. be *api-app.cognigy.ai*.

Natural Language Support As we have already mentioned in the document, our product has support for a wide variety of different languages in which you can build your bots. If you follow the instructions closely, we will only activate and deploy the microservices which are required to run our **English** NLU stack. If you don't need English NLU or want to add support for even more languages, you can do so by modifying the main **kustomization.yaml** file located under

[16]https://kubernetes.io/docs/concepts/services-networking/ingress-controllers/

core/development/product

Just search for the section with the comment *nlp deployments* and uncomment the languages you want to run in addition (or comment the services for the English languages in case you don't need it). 3.3 Activating additional support for Japanese and German - NLU languages shows an example of how you can activate Japanese and German language support in addition to English.

> Activating and deploying additional languages requires additional hardware (CPU, memory, etc.) in your Kubernetes cluster. Make sure that you have enough resources available in order to activate additional languages.

```
15  # nlp deployments
16  # for languages: de-DE
17  - manifests/nlp-deployments/de/service-nlp-score-de.yaml
18  - manifests/nlp-deployments/de/service-nlp-train-de.yaml
19
20  # for languages: en-US,en-AU,en-CA,en-IN,en-GB
21  - manifests/nlp-deployments/de/service-nlp-score-en.yaml
22  - manifests/nlp-deployments/de/service-nlp-train-en.yaml
23
24  # for languages: ge-GE,sv-SE,fi-FI,da-DK,nn-NO,vi-VN
25  # - manifests/nlp-deployments/de/service-nlp-score-ge.yaml
26  # - manifests/nlp-deployments/de/service-nlp-train-ge.yaml
27
28  # for languages: ja-JP
29  - manifests/nlp-deployments/de/service-nlp-score-ja.yaml
30  - manifests/nlp-deployments/de/service-nlp-train-ja.yaml
31
32  # for languages: ko-KR
33  # - manifests/nlp-deployments/de/service-nlp-score-ko.yaml
34  # - manifests/nlp-deployments/de/service-nlp-train-ko.yaml
35
36  # for languages: ar-AE,es-ES,fr-FR,nl-NL,it-IT,pl-PL,pt-PT,pt-BR,th-TH,ru-RU,tr-TR,zh-CN
37  # - manifests/nlp-deployments/de/service-nlp-score-xx.yaml
38  # - manifests/nlp-deployments/de/service-nlp-train-xx.yaml
39
```

Figure 3.3: Activating additional support for Japanese and German - NLU languages

Once you are done with patching the necessary files, we can finally deploy the Cognigy.AI application by using the appropriate command from our cheatsheet.

> Cheatsheet, 3.6 Installing Cognigy.AI, Deploying Cognigy.AI by using kustomize

Deploying the application will take some time as your Kubernetes cluster will have to download all container images from our container image registry first. If the bandwith of your server is fast enough, you should have your system running in about 20-25 minutes. You can monitor your deploying by using the command from our cheatsheet.

> Cheatsheet, 3.6 Installing Cognigy.AI, Monitoring the deployment

Once all deployments report as being available, you can open a web-browser and

visit the domain which you have choosen for our frontend application. You should be able to see the login screen of our product like in 3.4 Login screen of Cognigy.AI 4.

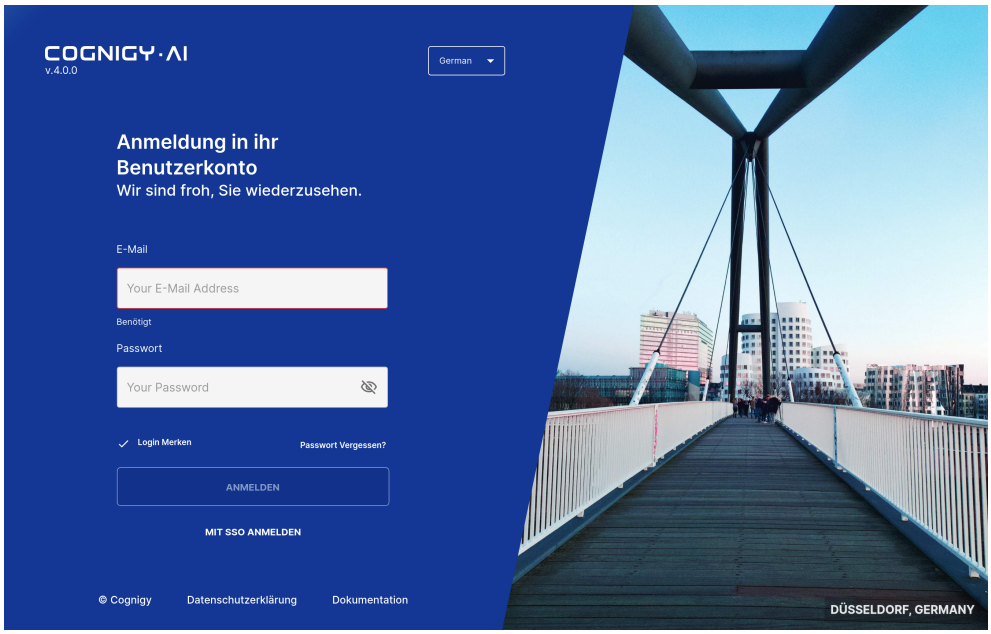

Figure 3.4: Login screen of Cognigy.AI 4

3.7 Retrieve login credentials

Once you are able to see the login screen of our product, you migth wonder which credentials can be used to login into the product. We have decided against adding a default *admin* user with static-credentials to our product as these sort of users will often not get properly removed from systems and leave software products exposed to potential attackers. Our system creates random credentials on startup if no credentials exist within the system, yet. The credentials will get printend into the log of one of our microservices - we can them by retrieving the log-outputs of our **service-security** microservice. Our cheatsheet contains the necessary command to retrieve the current log-statements of the specific Pod.

> Cheatsheet, 3.7 Retrieve login credentials, Retrieving credentials from service-security logs

3.5 Retrieving the auto-generated user-credentials in a Kubernetes based installation shows the log-statements of our security service including the randomly generated email and password you can use in order to login into the platform - after you have installed the product license-key.

```
-----------------------------------------------------------
[SECURITY] Connected to message broker and created channel...
This is the initial user account you can use to login
[SECURITY] Awaiting RPC requests on queue 'security-queue'...
email: user1d978235f9@cognigy.com
password: e84aa36075a72bc5ba8e93f40e591af9
```

Figure 3.5: Retrieving the auto-generated user-credentials in a Kubernetes based installation

3.8 Activating your license key

In 3.7 Retrieve login credentials we have covered how you can retrieve your initial login credentials. These are necessary in order to activate your license-key. You should have received your license-key together with the container image registry credentials which we have used in 3.2 Configuring container registry access.

Visit the web interface of Cognigy.AI, but replace the URL ending **/login** with **/license**. This will open our license activation screen which looks quite similar to the login-form - with the only difference being the license-key field. Insert email, password and your license-key and click on *Activate license*. If everything worked, the application will redirect to the login screen on which you can login to finally get started with using our product.

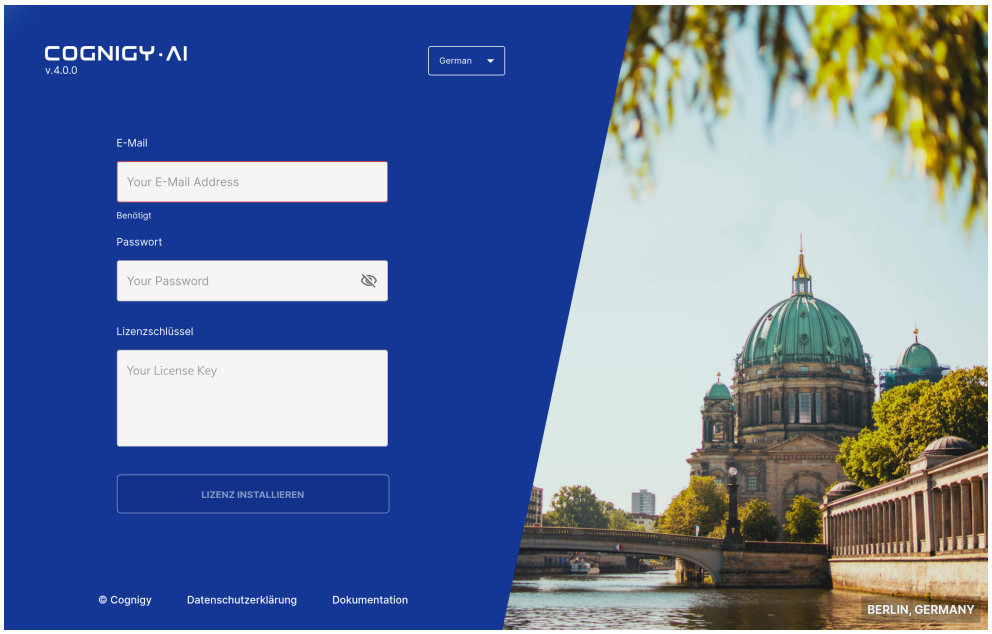

Figure 3.6: Activating your Cognigy.AI product license

Chapter 4

Configuration

4.1 Tweaks

Cognigy.AI is a highly configurable software product which was specifically designed to be tweaked by our customers. Our aim is that you have as much control over the software as possible - regarding its functionality, its features and its performance caracteristics. Configuring our software product is achieved by changing the values of a Kubernetes ConfigMap[1] which is located in your environment folder.

In the case of our example we are using throughout this manual, the file is located under **core/development/product/manifests/config-maps/config-map.yaml**. Please do not directly change this file as it is located in the manifests folder. Instead, add the necessary changes to the **config-map_patch.yaml** located in the **overlays/config-maps** directory. During the initial setup process of our software you should have already changed the contents of this file by chaning the referenced DNS names. The following pages of this manual describe all environment variables you can add.

Once you have modified the contents of your patch file, you have to apply the patches and re-deploy the application. You can use the very same command that we have already used to initialy deploy the product into your cluster - the cheatsheet contains the command for that.

> Cheatsheet, 3.6 Installing Cognigy.AI, Deploying Cognigy.AI by using kustomize

[1] https://kubernetes.io/docs/concepts/configuration/configmap/

Option	Description	Default	Microservices
FRONTEND_BASE_URL_WITH_PROTOCOL	This is the URL you use to access the application (e.g. https://ui.company.com)	-	service-api, service-ui
BACKEND_BASE_URL_WITH_PROTOCOL	This is the URL of the API of the application (e.g. https://api-ui.company.com)	-	live-agent-ui, service-api, service-ui
ENDPOINT_BASE_URL_WITH_PROTOCOL	This is the URL for the endpoint service (e.g. https://endpoint-ui.company.com)	-	live-agent-ui, service-alexa-management, service-api, service-ui, service-webchat
WEBCHAT_BASE_URL_WITH_PROTOCOL	This is the URL you use to access the demo webchat page (e.g. https://webchat-ui.company.com)	-	service-api, service-ui
ODATA_HOSTNAME	This is the URL you use to extract analytics via ODATA (e.g. https://odata-ui.company.com)	-	service-analytics-odata

Option	Description	Default	Services
MESSAGE_TTL_SECONDS	This variable specifies the TTL in seconds of messages sent by users. This will delete messages after the TTL has passed if they were not processed yet to make sure that new messages can be processed.	120 seconds	service-endpoint
MAX_MEMORY_OBJECT_SIZE	This variable specifies the maximum size the context can have in a Flow. The value is in bytes	65536 bytes	service-ai
HTTP_JSON_BODY_LIMIT	This variable specifies the maximum size the payload used in our REST API can have. The value is in bytes	65536 bytes	service-api
HTTP_NODE_TIMEOUT_IN_SECONDS=	This variable specifies the maximum amount of seconds an HTTP request to a third-party system can take until the product aborts the request. The value is in seconds and the default is 8 seconds.	Unspecified	service-ai
MAX_BYTE_SIZE	This variable specifies the maximum size the result of the database query from a DB node can have. The value is in bytes	500000 bytes	service-database-connections
RESPONSE_BYTES_LIMIT	This variable specifies the maximum size the HTTP response from the API node can have. The value is in bytes	524288 bytes. Can be set to 0 to remove the limit	service-http
RSS_FEED_URL	This variable configures the RSS Feed on the projects page.	It will per default load news about Cognigy.AI (e.g. release updates)	service-ui

Option	Description	Default	Services
LOG_ENTRIES_TTL_IN_MINUTES	This variable specifies when the logs stored in our product expire and get deleted. The value is in bytes.	65536 bytes	service-logs
LOG_ENTRIES_BUFFER_IN_SECONDS	This variable specifies how long to buffer log entries before storing them in the database.	5 seconds	service-logs
API_CORS_WHITELIST	This is a whitelist of domains that are allowed to access the API. If not specified, then all domains can access the API. This is a comma-separated list of urls, e.g. http://some-domain.com, http://exammple.com	Unspecified	service-api
ENDPOINT_CORS_WHITELIST	This is a whitelist of domains that are allowed to access the endpoint. If not specified, then all domains can access the endpoint. This is a comma-separated list of urls, e.g. http://some-domain.com, http://exammple.com	Unspecified	service-endpoint
ALEXA_END_SESSION_AFTER_EACH_REPLY	Whether to end a session after each reply on Alexa per default. This can be overwritten in a Say Node.	true	service-endpoint
EXTERNAL_ENDPOINT_BASE_URL_WITH_PROTOCOL	Adding the query parameter ?external=true to the webchat demo page will make it use this Endpoint URL.	Unspecified	service-webchat
WEBCHAT_FORCE_WEBSOCKETS	Per default we force websockets for certain browsers. Set this to false to disable this in case your proxy doesn't support websockets.	It will per default only force websockets for IE11 and Safari	service-webchat

Option	Description	Default	Services
MAX_LOOPS	This specifies the allowed amount of round-trips in the Flow-Editor. It is used for our infinite loop detection algorithm.	4	service-ai
MODULE_MAX_EVENT_EMISSIONS	This specifies how many actions a Custom Module can use in one execution.	10	
MAX_MODULE_EXECUTION_TIME_IN_SECONDS	The maximum amount of time a Custom Module can execute before it times out.	20 seconds	service-execution
EXPORT_IMPORT_ENCRYPTION_KEY	Custom encryption key for Import/Export packages. If you define this key, you will no longer be able to share packages with other environments!	Unspecified	service-export-import
MAX_CONTACT_PROFILE_TTL_IN_MINUTES	The maximum amount of time before a contact profile is deleted from the database. Setting the value to 0 means that they never expire.	0	service-profiles
MAX_CONVERSATION_TTL_IN_MINUTES	The maximum amount of time before a conversation is deleted from the database. Setting the value to 0 means that they never expire.	0	service-analytics-conversation-collector
MAX_SESSION_STATE_TTL_IN_MINUTES	The maximum amount of time before a session expires. Setting the value to 0 means that they never expire.	0	service-ai

Option	Description	Example value	Services
SECURITY_SMTP_HOST	The host of the SMTP server to use	smtp.office365.com	service-security
SECURITY_SMTP_PORT	The port used by the SMTP server	587	service-security
SECURITY_SMTP_CONNECTION_TYPE	Can either be "none", "starttls" or "tls"	"starttls"	service-security
SECURITY_SMTP_USERNAME	The username of the user sending the email	h.miller@cognigy.com	service-security
SECURITY_SMTP_PASSWORD	The password for the user sending the email. Should be configured as a secret in Kubernetes	-	service-security
SECURITY_SMTP_FROM	The email address which will get displayed to the user.	noreply@mail.com	service-security
FEATURE_USE_SMTP_FROM	Feature flag with which you can disable an additional FROM field for SMTP connections in our product.	Unspecified/false	service-security

4.2 Custom theme

4.2.1 Overview

Cognigy.AI offers the ablity to customize the look and feel of its user interface through what we call a *custom theme*. A custom theme can contain the following assets:

- logo.png
- logo_primary.png
- logo_square.png
- main_background.png
- static_content1.png to static_content6.png
- custom_config.json
- favicon/
  - android-icon-192x192.png
  - apple-icon-57x57.png
  - apple-icon-60x60.png
  - apple-icon-72x72.png
  - apple-icon-76x76.png
  - apple-icon-114x114.png
  - apple-icon-152x152.png
  - apple-icon-180x180.png
  - favicon-16x16.png
  - favicon-32x32.png
  - favicon-96x96.png
  - favicon-256x256.png
  - favicon.ico
  - ms-icon-70x70.png
  - ms-icon-144x144.png
  - ms-icon-150x150.png
  - ms-icon-310x310.png

If you plan to apply a custom theme to your installation as well, please don't hesitate to send an email to **support@cognigy.com**. We will provide you with an example theme you can use as a basis for your own theme.

4.2.2 Asset requirements

logo.png The logo will be displayed on the login-screen as well as on various pages within our products user interface. Please ensure that this asset:

- uses the PNG file format - transparency is supported
- must have a height of 24 pixels
- should have width of 194 pixels, although the width is dynamic

logo_primary.png This variant of the logo will be used if the logo gets rendered on the primary color of your theme. In our case, our variant of this logo is white as our primary color is a dark blue. Please ensure that this asset:

- uses the PNG file format - transparency is supported
- must have a height of 24 pixels
- should have width of 194 pixels, although the width is dynamic

logo_square.png This variant of the logo has a small square format. It will be used in our sidebar - the navigation within an agent. Please ensure that this asset:

- uses the PNG file format - transparency is supported
- must have a resolution of 24 x 24 pixels
- should have a good contrast on your primary color (dark blue in our case)

main_content.png This image will be shown on the login page and will be zoomed and clipped to fill the available space. The image will get overlapped by parts of the login form.

- uses the PNG file format
- should have a tradeoff between image quality and image size for fast page loads
- should have a resolution of 1920 x 1080 pixels
- you can also e.g. use twice the resolution if you want to support high-dpi devices and screens

static_content1.png - static_content6.png The static content images will be displayed on our dashboards. They will shrink in width on shallow viewports and will be clipped to cover space.

- uses the PNG file format
- should have a tradeoff between image quality and image size for fast page loads
- should have a resolution of 418 x 222 pixels
- you can also e.g. use twice the resolution if you want to support high-dpi devices and screens

4.2.3 How to build a theme

The main configuration for a theme is stored in a file called **custom_config.json**.

Changing colors To change the colors of the application, refer to the *theme* section.

Theme buttons and interaction elements To change the colors of interaction elements, change the values for *theme.cognigy.functionColors.primary*, its siblings as well as the *theme.palette.primary.main* and *theme.palette.sencondary.main* values.

Theme the sidebar To change the colors of elements within the sidebar, refer to the values set in *theme.cognigy.functionColors.sideBar*.

Defining static content You will have to provide custom content which will be rendered on the index dashboard as well as agent dashboards. The *get more* link for the static content can be changed by modifying *links.staticContentLink*.

Index dashboard static content To change the static content for the index-dashboard, update the content of *staticContentCards*. You can link to theme-provided images by setting the *imageUrl* to */custom/theme/someImageInYourThemeFolder.jpg*.

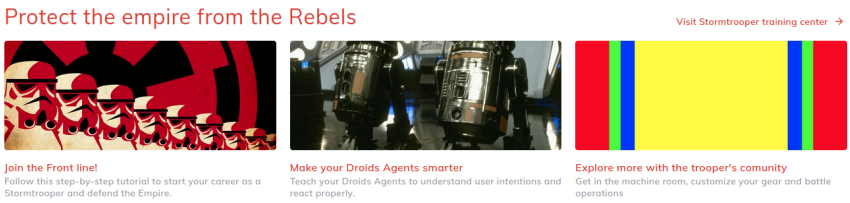

Figure 4.1: Static content cards

To define the index dashboards cards, exchange the content of *staticContentCards*, for the agent dashboards, refer to *agentDashboardStaticContentCards*. You can change

the title of the section with *staticContentTitle* and customize the link with *staticContentLinkTitle*.

Agent dashboard static content Follow the same steps as for the index dashboard static content, just edit:

- *agentDashboardStaticContentCards*
- *agentDashboardStaticContentTitle* and
- *agentDashboardStaticContentLinkTitle*

Newfeed The news feed will be shown when a URL was provided in the *RSS_FEED_URL* field in your main configmap. Currently, the default value is pointing to one of our Cognigy RSS feeds. When no URL is provided, the static content will grow to occupy the whole layout. A maximum of 6 cards will be displayed instead.

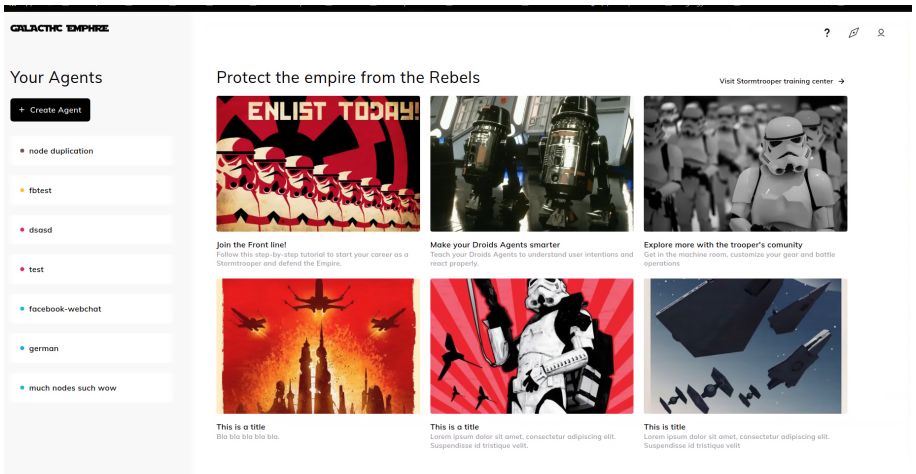

Figure 4.2: The application can display up to six static content cards.

When the newsfeed is active, there will be a maximum of 3 static content cards and 3 news cards displayed.

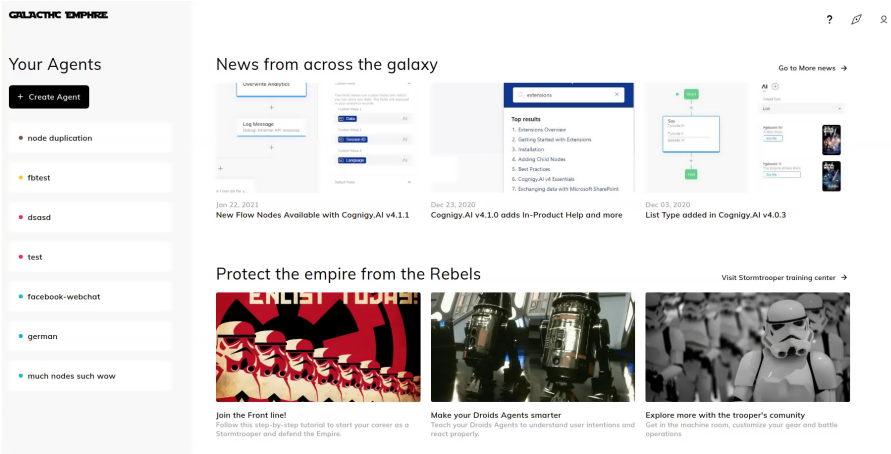

Figure 4.3: The newsfeed as well as three static content cards.

4.2.4 Configuring a theme

In order to apply a custom theme to your own Cognigy.AI installation, you essentially have to create the individual assets of the theme as Kubernetes config-maps. Please follow the commands within the cheatsheet in our kubernetes repository.

> Cheatsheet, 4.2 Custom theme, Configuring a theme

Once you have created all configmaps in your cluster, we have to add a patch-file which will modify your *service-ui* deployment whenever you apply this manifest to your cluster. We will rely on the folder structure of the kubernetes repository and create a new *service-ui_patch.yaml* file which will apply the necessary changes to your UI deployment.

Please create a new folder in your kubernetes repository, more specifically in **kubernetes/core/development/product/overlays**. Name this folder **deployments** and create a new file called **service-ui_patch.yaml** within it. Fill the file with the contents from our cheatsheet.

> Cheatsheet, 4.2 Custom theme, service-ui patch to apply the theme

Lastly you have to modify your **kustomization.yaml** in the product folder as we also have to make sure that our additional patch-file will be loaded during the deployment. Open the file in a text-editor and add the couple of lines from our cheatsheet right before the **patchesStrategicMerge** block.

> Cheatsheet, 4.2 Custom theme, Loading your additional patch

If you are now running the next deployment of our product, your service-ui deployment will receive a couple of new key/value pairs and will load your custom theme. If you have any questions or encounter any problems while applying your custom theme, please raise a support ticket by sending an email to: **support@cognigy.com**.

4.3 CORS configuration

CORS[2] is a security feature implemented in modern web-browsers and allows additional protection for your applications. Web-browsers will send so-called preflight HTTP requests to web-servers in order to retrieve a list of allowed CORS origins. If the provided list of allowed origins does not contain the domain from which the actual request should be sent, the browser will deny to send the request.

Cognigy.AI shipps with CORS configuration support for the integrated RESTful API. As of Cognigy.AI version 4.4, we have also added support for configuring CORS origins for the endpoint. In order to configure CORS you can use the following additional environment variables:

Listing 4.1: Environment variables to configure allowed CORS origins

```
# allowed CORS origins for 'service-api'
API_CORS_WHITELIST=https://origin1.domain.com,https://example.com

# allowed CORS origins for 'service-endpoint'
ENDPOINT_CORS_WHITELIST=https://chat.example.com
```

By default, Cognigy.AI allows all origins which will set the list of allowed origins to '*'. As you can see in 4.1 Environment variables to configure allowed CORS origins you simply add the list of allowed origins and separate them by comma. In order to get more instructions on how you can actually set these environment variables in a persistent way, please follow the steps described in 4.1 Tweaks

[2]CORS = Cross Origin Resources Sharing

Chapter 5

Updating

5.1 Preparation

Updating a Cognigy.AI installation is easy based on the new tooling with kustomize and the Git repository. Once we have released a new version of our software, please first go ahead and carefully read through our release-notes[1] in our product documentation. Please also carefully go through the **changelog**[2] in the kubernetes repository in order to understand the changes we might have implemented regarding the manifest files in the kubernetes repository.

5.2 Retrieving the update

Once you are fully aware of all changes the new release might bring in, you can pull the latest changes from GitHub and checkout the new version[3] Our cheatsheet contains the commands you can use.

> Cheatsheet, 5.2 Retrieving the update, Make local repository current

If pulling the latest changes was successful, you can go ahead and use the same command which we have used when changing the configuration of your system or during the initial installation. Running the kustomization command and applying the resulting manifests will make sure that your overlays and patches properly adjust certain aspects of the new release.

> Cheatsheet, 3.6 Installing Cognigy.AI, Deploying Cognigy.AI by using kustomize

[1] Cognigy.AI release notes, https://docs.cognigy.com/docs/release-notes

[2] Changelog of our Kubernetes repository, https://github.com/Cognigy/kubernetes/blob/master/CHANGELOG.md

[3] Cognigy creates tags in the Git repository for each release.

Chapter 6

Cognigy Live Agent

Cognigy Live Agent was introduced with our Cognigy.AI 4.1.0 release and is currently only functional when you are also using our core product Cognigy.AI. Please ensure that you have a fully working Cognigy.AI installation before you have a look at this section of our documentation. Please also refer to our Cognigy Live Agent documentation located on https://docs.cognigy.com.

Please note that Cognigy Live Agent, released with Cognigy.AI 4.1.0, is currently in an early alpha. If you experience issues, instabilities or have other ideas for improvements, please get in touch with us and send us an email to support@cognigy.com. Please also note that you need to have Cognigy.AI 4.1.0 installed in order to use Cognigy Live Agent.

6.1 Prerequisites & Hardware requirements

In order to set up a Cognigy Live Agent installation, check the following list of requirements before you get started:

- **Already existing Cognigy.AI installation**: Installing the Cognigy Live Agent requires you to have an already existing Cognigy.AI installation on version 4.1.0 and up.
- **Domains / DNS names**: You will also need two additional DNS names for two additional web-services. Please ensure that you have access to services which are required to create those DNS entries (e.g. a platform to manage DNS entries for your domain(s) like Go-Daddy, United-Domains etc.). If you only want to install the Cognigy Live Agent available to a private network, please ensure that you can create DNS entries for this private network.

The Cognigy Live Agent in its current form is lightweight and needs to following additional hardware in your Kubernetes cluster:

Services	Resource limits
Cognigy Live Agent (UI and API)	0.2 CPU, 150 MB
Total	0.2 CPU, 150 MB

6.2 Installation

6.2.1 Introduction

Installing Cognigy Live Agent re-uses concepts which you have already used while installing our main product, Cognigy.AI. Please ensure that you understand the installation procedure in 3.1 Cognigy.AI manifest files. All Live Agent related files are also stored in our Kubernetes repository and contained within a **live-agent** folder.

Similar to what we have described in the core product installation, we keep the actual **manifests** separate to your **overlays**. This allows us to ship changes to the original manifests without breaking your custom configuration. You will find the following structure in the **live-agent** directory:

- **manifests**
  Contains our original manifests which are necessary to deploy the Cognigy Live Agent application. These files can change in future releases - so please don't change them as your changes will be overwritten when you pull in the changes for a future release.
- **template.dist**
  Contains a blueprint of patch-files in a kustomize-compatible format. The folder mirrors the directory-structure which you can find in the manifests folder. Please also leave these files untouched as they only act as a blue-print. We will also update these files with future releases.
- **make_environment.sh**
  A script which allows you to make a copy of our *template.dist* folder as well as update a copy of the *manifests* folder within your template-copy. This script will be used to generate files for your Cognigy Live Agent installation.

> Customers can use the make_environment.sh script in order to maintain configuration for multiple Live Agent installations. You might set up multiple Cognigy.AI installations already - one for development, one for staging and one for production. Your Cognigy.AI setup can be mirror for Cognigy Live Agent.

6.2.2 Using templates

Let us now use the power of the templating approach in order to prepare your own files for your Cognigy Live Agent installation.

Open a terminal, navigate to your local copy of our kubernetes repository, make the *make_environment.sh* script executable and create a folder for your Cognigy Live Agent environment by executing the script with a name for your environment - we will pick **development** for this example.

> Cheatsheet, 6.2.2 Using templates, Preparing the files for a Cognigy Live Agent installation

After you ran the *make_environment.sh* script with the suffix for your enviornment, you should be able to see a new folder containing the file-structure we will use for our further work. The contents in this directory will remain stable even if you pull the latest changes from GitHub.

6.2.3 Secrets

Cognigy Live Agent also uses secrets in order to store database credentials in a secure way. You can read more about secrets in 3.3 Secrets. Once you ran the *make_environment.sh* script and have generated a folder for your environment, you will find a *secrets.dist* subfolder in that directory. We now have to create a copy of *secrets.dist*, create random credentials and apply our secrets to the cluster.

> Cheatsheet, 6.2.3 Secrets, Preparing the Live Agent secrets for your installation.

Once you have created the copy of the *secrets.dist* folder, we can now generate random credentials (we will need to re-use them later on) and apply your secrets to the cluster.

> Cheatsheet, 6.2.3 Secrets, Generating safe values for your Live Agent secrets.

Once these commands ran successfully, you should have two secrets located in the *secrets* folder which contain base64-encoded random credentials - and you have applied the changes to your Kubernetes cluster.

6.2.4 Storage

Live Agent reuses the storage systems that our main product Cognigy.AI uses as well. Hence the steps regarding storage are relatively simple - we essentially only need to create a new database in our **mongo database server**.

> Cheatsheet, 6.2.4 Storage, Creating additional database and user for Live Agent.

6.2.5 Patch files

The last step is to change a couple of *overlays* and adjust them to your needs accordingly.

Live Agent Config Map Cognigy Live Agent uses a separate configmap to store its configuration and environment variables. The patch-file you need to modify is located under:

live-agent/development/overlays/config-maps

You have to patch all fields with the appropriate values. Some of the fields are related to your Cognigy.AI installation. Please have a look at your Cognigy.AI patch-files in order to get their values. Please specifically have a look at the two last values in this patch file as they are crucial for a working Cognigy Live Agent setup. These values also have to match to values which we still have to add to the main Cognigy.AI configmap! 6.1 Adjusting the Live Agent ConfigMap patch shows an example for a valid configuration.

```
# DNS names
- op: add
  path: /data/LIVE_AGENT_BACKEND_BASE_URL_WITH_PROTOCOL
  value: "https://live-agent-api-staging-v4.cognigy.ai"

# DNS name of your Cognigy.AI API.
- op: add
  path: /data/BACKEND_BASE_URL_WITH_PROTOCOL
  value: "https://api-staging-v4.cognigy.ai"

# DNS name of your Cognigy.AI UI.
- op: add
  path: /data/FRONTEND_BASE_URL_WITH_PROTOCOL
  value: "https://staging-v4.cognigy.ai"

# DNS name of your Cognigy.AI endpoint.
- op: add
  path: /data/ENDPOINT_BASE_URL_WITH_PROTOCOL
  value: "https://endpoint-staging-v4.cognigy.ai"

- op: add
  path: /data/LIVE_AGENT_CLIENT_SECRET
  value: "hZ8nu5CCJuLr9CSQtzRgNsXxsGM9ZNSS"

- op: add
  path: /data/LIVE_AGENT_API_SECRET
  value: "jgfYE7tUp87aGqAbXqhKQ6B7qfTLmcWh"
```

Figure 6.1: Adjusting the Live Agent ConfigMap patch

Cognigy.AI Config Map Cognigy Live Agent depends on Cognigy.AI and e.g. uses it for authentication purpose. In order to ensure that the communication and the authentication flow can work between both products, we also have to adjust the Cognigy.AI configuration. If you have followed our documentation closely, you should only have to patch a file that is located under:

core/development/product/overlays/config-maps

Open the file in a text-editor and append the three patch statements shown in ref-fig:liveAgentCognigyPatch Extending the Cognigy.AI ConfigMap patch in order to have a functional Cognigy Live Agent to the end of your Cognigy.AI config-map patch.

```
# Cognigy.AI Live Agent
- op: add
  path: /data/LIVE_AGENT_CLIENT_SECRET
  value: "<same-value-as-in-live-agent-env>"
- op: add
  path: /data/LIVE_AGENT_API_SECRET
  value: "<same-value-as-in-live-agent-env>"
- op: add
  path: /data/LIVE_AGENT_BACKEND_BASE_URL_WITH_PROTOCOL
  value: "<same-value-as-in-live-agent-env>"
```

Figure 6.2: Extending the Cognigy.AI ConfigMap patch in order to have a functional Cognigy Live Agent

Ingress objects You have to adjust the patches for your Cognigy Live Agent ingress routes. The patch-files you need to modify are located under:

live-agent/development/overlays/reverse-proxy/ingress

You have to patch all files and replace the already existing placeholders (e.g. *live-agent-api*) with your actual DNS names.

Once you are done with patching all necessary files, we can finally deploy the Cognigy Live Agent application by using the appropriate commands from our cheatsheet.

> Cheatsheet, 6.2.5 Patch files, Applying Cognigy Live Agent files in order to initiate initial deployment.

Please also ensure that you apply the changes we made to the patch files for the core Cognigy.AI product as described in our cheatsheet.

> Cheatsheet, 6.2.5 Patch files, Updating Cognigy.AI configuration and ensure that services have correct configuration.

Chapter 7

Cognigy Management UI

Cognigy Management UI is a lightweight companion product which allows our customers to manage their Cognigy.AI v4 systems. It allows our customers to create new organizations, create users, inspect billing relevant information and perform other maintenance and configuration operations like configuring the password-policy. Please note that you need to have a fully working Cognigy.AI setup in order to use the Cognigy Management UI. Please also refer to our Cognigy Management UI documentation located on https://docs.cognigy.com.

You need only one Cognigy Management UI deployment to be able to manage each of your Cogngiy.AI systems - in case you have multiple - where you have configured Cognigy Management UI access credentials.

7.1 Prerequisites & Hardware requirements

In order to set up a Cognigy Management UI installation, check the following list of requirements before you get started:

- **Already existing Cognigy.AI installation**: Installing the Cognigy Management UI requires you to have an already existing Cognigy.AI installation.
- **Domains / DNS names**: You will also need one additional DNS name for an additional web-service. Please ensure that you have access to services which are required to create those DNS entries (e.g. a platform to manage DNS entries for your domain(s) like Go-Daddy, United-Domains etc.). If you only want to install the Cognigy Management UI available to a private network, please ensure that you can create DNS entries for this private network.

The Cognigy Management UI in its current form is lightweight and needs to following additional hardware in your Kubernetes cluster:

Services	Resource limits
Cognigy Managment UI	0.02 CPU, 30 MB
Total	0.02 CPU, 30 MB

7.2 Installation

7.2.1 Introduction

Installing Cognigy Management UI re-uses concepts which you have already used while installing our main product, Cognigy.AI. Please ensure that you understand the installation procedure in 3.1 Cognigy.AI manifest files. All Management UI related files are also stored in our Kubernetes repository and contained within a **management-ui** folder.

Similar to what we have described in the core product installation, we keep the actual **manifests** separate to your **overlays**. This allows us to ship changes to the original manifests without breaking your custom configuration. You will find the following structure in the **management-ui** directory:

- **manifests**
  Contains our original manifests which are necessary to deploy the Cognigy Management UI application. These files can change in future releases - so please don't change them as your changes will be overwritten when you pull in the changes for a future release.
- **template.dist**
  Contains a blueprint of patch-files in a kustomize-compatible format. The folder mirrors the directory-structure which you can find in the manifests folder. Please also leave these files untouched as they only act as a blue-print. We will also update these files with future releases.
- **make_environment.sh**
  A script which allows you to make a copy of our *template.dist* folder as well as update a copy of the *manifests* folder within your template-copy. This script will be used to generate files for your Cognigy Management UI installation.

7.2.2 Using templates

Let us now use the power of the templating approach in order to prepare your own files for your Cognigy Management UI installation.

Open a terminal, navigate to your local copy of our kubernetes repository, make the *make_environment.sh* script executable and create a template folder for your Cognigy Management UI deployment by executing the script.

> Cheatsheet, 7.2.2 Using templates, Preparing the files for a Cognigy Management UI installation

After you ran the *make_environment.sh* script, you should be able to see a new folder **template** containing the file-structure we will use for our further work. The contents in this directory will remain stable even if you pull the latest changes from GitHub.

7.2.3 Secrets

Cognigy Management UI itself does not use any secrets, instead you need to configure the access secrets for each individual Cognigy AI deployment you want to manage trough the Management UI. You can read more about secrets in 3.3 Secrets. After updating to Cognigy AI version 4.12.0 you should see a new secret file in your environment folder **kubernetes/core/*environment*/product/secrets/cognigy-management-ui-creds.yaml**.

Figure 7.1: Adjusting the Management UI secret file

The secret has **JSON** format consisting of an **Array** holding multiple

{"username":"", "password":""}

pairs. To add multiple users, simply add one new pair of credentials per individual user.

> Cheatsheet, 7.2.3 Secrets, Preparing the Management UI secrets for your Cognigy AI deployment.

Once you have added the new credentials you can apply the changes to your Cognigy AI deployment.

> Cheatsheet, 7.2.3 Secrets, Applying the changes to you Cognigy AI deployment

7.2.4 Patch files

The last step is to change an *overlays* file and adjust it to your needs accordingly.

Ingress objects You have to adjust the patch for your Cognigy Management UI ingress route. The patch-file you need to modify are located under:

management-ui/template/overlays/

You have to patch the file and replace the already existing placeholder (e.g. *management-ui*) with your actual DNS name.

Once you are done with patching the necessary file, we can finally deploy the Cognigy Management UI application by using the appropriate commands from our cheatsheet.

> Cheatsheet, 7.2.4 Patch files, Applying Cognigy Management UI files in order to initiate initial deployment.