Implementation Guide

LumenVox Containers Release 4.6 (Updated 01/28/2024)

The information contained herein is proprietary and confidential and cannot be disclosed or duplicated without the prior written consent of The LumenVox Corporation.

Copyright © 2012–2024 The LumenVox Corporation. All rights reserved.

Containerized Architecture 7
Architecture Summary 7
APIs 8
Required External Software 8
RabbitMQ – Message Queuing 8
Redis – In-Memory Cache 9
MongoDB – Database 9
PostgreSQL – Database 9
Persistent Storage 10
Other Recommended External Components 10
Logging Tool 10
Monitoring Tool – such as Prometheus 10
Access and User Management 10
Licensing Service 11
Resources 11
LumenVox Containerized Microservices 11
Admin-portal 12
Archive 12
Asr 12
Audit 12
Binary Storage 13
Biometric-active 13
Biometric-api 13
Biometric-identity 13
Configuration 13
Deployment 13
Deployment-portal 14
Grammar manager 14
ITN 14
License 14
LumenVox-api 14
Management-api 14
Persistent-volume-directory-setup 14

© 2024 LumenVox

Reporting 14
Reporting-api 15
Reporting-bio-api 15
Resource 15
Session 15
Transaction 15
Tts 15
Vad 15

Communication Protocols 17
Supported Standards to Access LumenVox Services 17
gRPC 17
MRCP 17
REST 19

Administration & Maintenance Tools 20
Admin Portal & Deployment Portal 20
Analysis Portal 21

Kubernetes Installation Steps 21
Troubleshooting 25
Potential Installation pain-points 25

Logging 26

Monitoring 26

Licensing 26

Resources 27

APPENDIX 1 – Setup in Google GCP (GKE Creation) 28
Create Kubernetes Cluster 28
Set up cluster basics 28
Set up Node pool details 29
Set up node configurations 30
Set up node security 31
Set up node networking 32

© 2024 LumenVox

Create VPC Firewall Rule for Rabbit MQ 33
Add persistent storage 34
Install GCloud CLI 34
Install Linkerd CLI 35
Install Linkerd 35
Install Jaeger 35
Install Linkerd Dashboard 35
Uninstall Linkerd 35
Install Helm 35
Install nginx Ingress 36
Create LumenVox Namespace 36
Change NameSpace 36
Setup TLS for Ingress 36
Apply Secrets File 36
Installation of Helm Charts 36
Deploy Public Chart 36

APPENDIX 2 – Amazon Kubernetes Service (EKS) 37
Create EKS Cluster 37
EKS Cluster Networking 38
Select Cluster Endpoint Access 39
Networking add-ons 40
Configure logging 41
Create node group and create node 41
Configure Node Group 42
Set compute and scaling configuration 43
Specify Networking 46
Create EFS Instance (elastic file storage) 46
Network Access 47
Install kubectl 47
Create access keys 47
Install AWS CLI 48
Update kubeconfig 48
Install Helm 48

Install Linkerd CLI....48
Install Linkerd....49
Install Jaeger....49
Install Linkerd viz Dashboard....49
Uninstall Linkerd....49
Install nginx Ingress....49
Create LumenVox Namespace....49
Change NameSpace....49
Setup TLS for Ingress....50
Apply Secrets File....50
Configure Helm....50
Deploy Public Chart (Internal Resources)....50

APPENDIX 3 – LV Containers Quick Start (kubeadm)....51
Getting started....51
Install LumenVox Containers Repo....51
Generate SSL Certificate....51
Grant execute permissions to scripts....51
Perform Installation....51

APPENDIX 4: Setting up a deployment....55
Configure set up of deployment information....55
Create deployment....55

APPENDIX 5 – Basic KUBECTL Commands....57
Get list of pods....57
Get status of each pod....57
Get logs for a pod....57
Show ingress....57
To remove a pod....57
To scale a pod....57

APPENDIX 6 - Sample secrets file....58

APPENDIX 7 - MRCP API Server....59
Installation....59

© 2024 LumenVox

Server Ports Setup 60
Firewalls 62
Cluster GUID 62

About LumenVox 68

Document Overview

This document provides detail on the implementation and management of the LumenVox Containers product. It also provides further detail on the various components of the product along with pre-requisites required for installation. The document supplies installation steps that can be followed for Google and Amazon Kubernetes cloud environments as well as on-premises installation using the Quick Start installation process (using kubeadm).

Product Summary

LumenVox Containers utilizes state of the art container technology to implement our speech and voice biometrics solutions via microservices architecture. These containers can be deployed using Kubernetes either on-premise, within a cloud/multi-cloud environment or via a hybrid model.

LumenVox allows for clients to integrate their web frontends, IVR, smartphone applications, Chatbots, conversation AI applications or backend systems into LumenVox's technology.

Using LumenVox's new containerized solution offers the following benefits:

- Auto scaling - this allows the client to easily increase or decrease the number of containers that are running to meet capacity needs when workload changes. This can be done by setting parameters (e.g., Peak Load Times). By default, it is recommended that this is turned off if your technology is hosted so that you don't consume infinite resources. Please liaise with LumenVox before enabling this.
- Self-healing – using Kubernetes, containers can be easily replaced should one fail. This also assists with version rollbacks and makes upgrades easier.
- A client can easily create a local or global installations
- Fail-over / disaster recovery (DR) – is made possible using the containerized technology
- Simple deployment – with the use of Helm charts
- Easier to integrate across the LumenVox products stack e.g., ASR with Transcription, text-to-speech (TTS), active & passive voice biometrics, Call Progress Analysis (CPA).

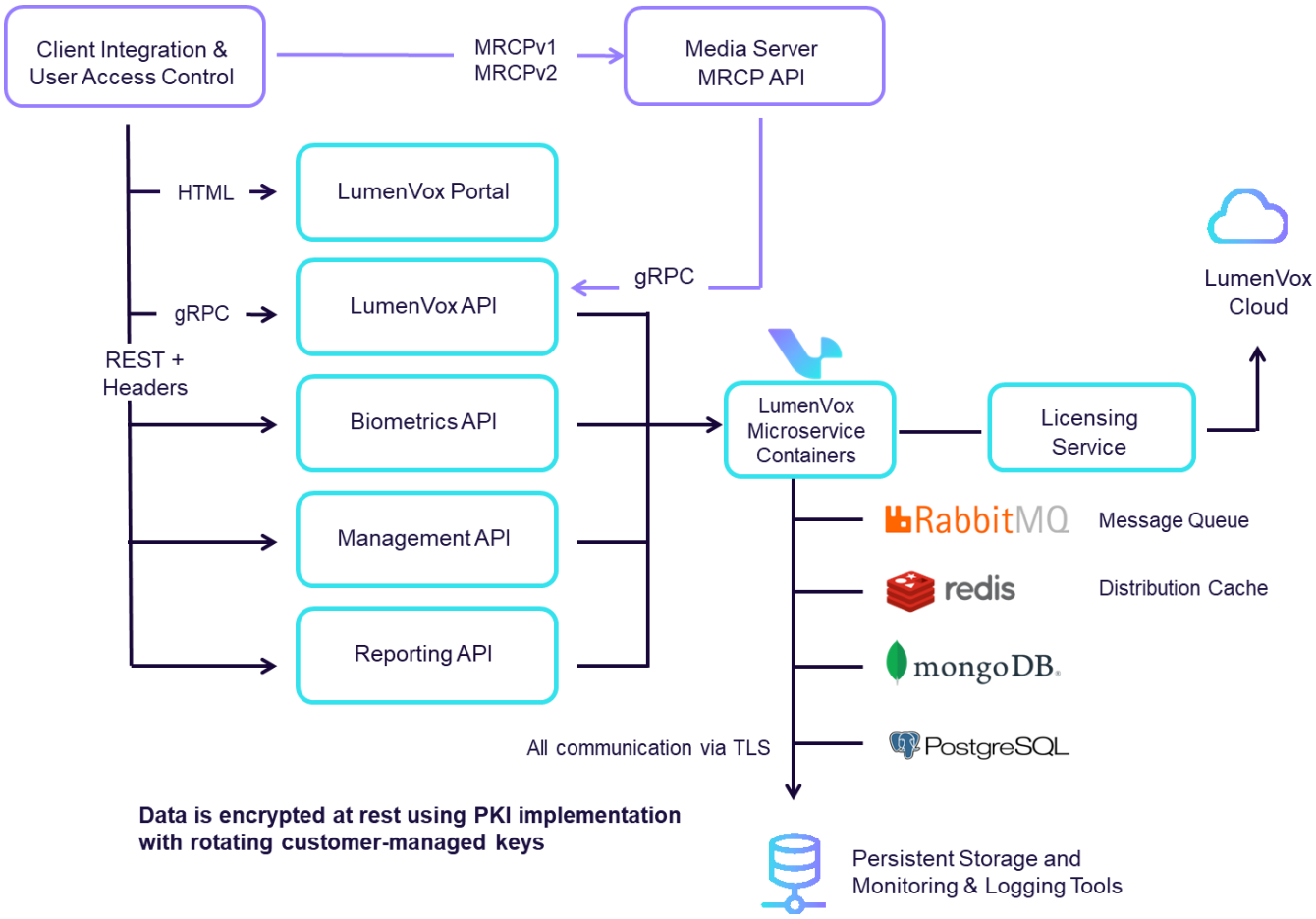
Further information on the LumenVox container and the products available can be found by accessing the LumenVox Containers Speech and LumenVox Containers Active Voice Biometrics product guides

© 2024 LumenVox

Containerized Architecture

Architecture Summary

The following diagram depicts the high-level architecture of our cloud-native containerized software. It shows how the client can integrate with LumenVox speech products through APIs.



In the following sections the different components of this diagram are discussed in more detail:

1. APIs
2. Required external software (including Persistent Storage)
3. Other recommended external components (monitoring, logging, and access control)
4. Licensing Service
5. LumenVox Containerized Microservices

The LumenVox admin portal and the communication protocols are covered in later sections.

© 2024 LumenVox

APIs

These public APIs are APIs that the LumenVox Containers expose to enable applications to utilize the software services provided by LumenVox.

- **LumenVox API** – the LumenVox API is used for functionality working with LumenVox speech products. This collection of APIs can be used to consume the speech products: ASR, Transcription, TTS, CPA & AMD.
- **Biometrics API** – this collection is used for LumenVox Voice Biometrics products. It is used to perform active voice biometrics enrollment and verification transactions.
- **Management API** – REST API that provides external access to the Deployment, Configuration and Engine Resource Services. Used to manage configuration and deployment parameters, which can also be managed via the Admin Portal and the Deployment Portal.
- **Reporting API** – this collection of APIs is used to extract session and transactional data from the platform. There is currently a Reporting API for LumenVox Speech, and a Reporting Bio API for LumenVox voice biometrics. The Reporting API for Speech is a REST API that connects via gRPC to the Reporting service and databases.

For more information on the APIs, and how client applications can interact with them, visit our API documentation at https://developer.lumenvox.com/

Required External Software

The following external software components are required and must be supplied & supported by the client (including monitoring, backups, etc). The first four components have been selected by LumenVox as the required solutions for complete functioning of our software in the containerized architecture. They are available as open-source, or as cloud-managed by major public cloud providers. Each of these are optimal and best-in-class. Partners or clients are required to implement these as a pre-requisite.

In a production environment, a distributed cluster of each of these components is recommended, as opposed to a singular instance, to ensure high availability, scalability, and disaster recovery.

RabbitMQ – Message Queuing

This message queuing component, to be implemented independently by the client/partner, is used by LumenVox software for all internal messaging between the microservices, which means the queuing of all transaction, audit, and management information. Once a service processes the data requests, they are removed from the queue.

Version: 3.9.16 (recommended)

© 2024 LumenVox

Redis – In-Memory Cache

This in-memory data structure store acts as a database Cache. It holds the session IDs, audio IDs and Interaction IDs for the audio processing transaction. It is used for caching of session & transaction information throughout the architecture for the duration of its processing by the various internal services. It is this component that receives and processes the audio for speech processing.

Version: 7.0.13 (minimum)

MongoDB – Database

A Key-Value based database is used as a repository for session binary data. This document-oriented database is used by LumenVox software to store the audio, grammars, and SSML files in a binary format. It is important to store this information for audit/troubleshooting purposes. It is used to determine and refine the performance of the product, using the Analysis portal described later. The storing of audio files is configurable. Clients have the ability to customize the MongoDB database name (excluding KubeAdm deployments).

Version: 5.0.16 (minimum)

Storage requirements

TTS: 2.5kb MongoDB storage per minute

ASR and Transcription: 500kb MongoDB storage per minute

PostgreSQL – Database

A repository for persistent metadata, licensing, and reporting data. This SQL compliant relational database is used by LumenVox software to store all the transactional data. The system makes use of a single PostgreSQL database and the name can be customized by the client (excluding KubeAdm deployments).

Version: 13 (recommended)

Storage requirements

TTS:2.5kb Postgres storage per minute

ASR and Transcription: 25kb Postgres storage per minute

© 2024 LumenVox

Persistent Storage

In addition to the required pre-requisite components specified above, clients should also provide persistent storage (100GB SSD recommended - but will be dependent on client sizing), from a provider of their choosing. This is required for storing ASR & TTS models as well as grammar files. This will increase as different language models for ASR, TTS and Voice Biometrics are added. The grammar cache also resides within this volume so will increase as and when different grammar files are used. The size of the grammar files used will also have an impact on usage.

Other Recommended External Components

Logging Tool

Customers can supply their own logging platform to retrieve and analyze container logs. Customers are not restricted in which logging tools or frameworks they wish to use when working with K8s. LumenVox supports the common containerized logging architecture, allowing customers to select whichever log consolidation or filtering application they choose.

Cluster logging levels can be set to information, warning, error, debug - this is done in the values.yaml file and is applied to the entire system. This can be overwritten at a deployment level using the portal available.

Monitoring Tool – such as Prometheus

Prometheus is a graphical interface tool commonly used by cloud providers for monitoring overall system performance (e.g., CPU status). You can set up alerts to send messages or emails and access its dashboard at a pre-configured URL. However, any graphical interface cloud monitoring tool can be used. Another commonly used tool is Grafana. Any popular tool should work just fine.

Prometheus has the advantage that there are managed versions of the Prometheus service available from most of the main cloud hosting providers, such as Amazon, Google, and IBM.

LumenVox has out-of-the-box support for Prometheus. Prometheus metrics are available for each of the containers.

Access and User Management

Customers must provide their own access management Interface. LumenVox does not provide any user identification (authentication) or permission checking (authorization) for access to the

© 2024 LumenVox

APIs used to access the containerized products, this is managed by security systems the client will have in place.

Licensing Service

To consume LumenVox products, the containers are required to communicate with LumenVox's cloud licensing service to submit information on product utilization. This outbound connection is made using secure HTTPS, typically once per day, and contains no user identifiable information (only license usage metrics).

Customers need to ensure that external firewall requirements are modified to allow the external connection.

See licensing section below for more information for the opening of firewalls.

Resources

To consume LumenVox products, the containers are required to communicate with LumenVox's resources service to download speech or voice biometric models required for the product utilization. Customers need to ensure that external firewall requirements are modified to allow the external connection.

See resources section below for further information for the opening of firewalls.

LumenVox Containerized Microservices

At the heart of the LumenVox modern, cloud-native offering are the microservices in a containerized architecture.

This section provides insight into the various LumenVox containers which can be used for troubleshooting purposes.

The list of available of basic pods (containers) is:

- Admin-portal
- Archive
- Asr
- Audit
- Binary Storage
- Biometric-active
- Biometric-api
- Biometric-active

- Biometric-identity
- Configuration
- Deployment
- Deployment-portal
- Grammar
- ITN
- License
- LumenVox-api
- Management-api
- Persistent-volume-directory-setup
- Reporting
- Reporting-api
- Reporting-bio-api
- Resource
- Session
- Transaction
- Tts
- Vad

Admin-portal

This service manages the web portal for the cluster admin and supports the setup of each licensed tenant with their connection strings, encryption, and passwords to services and databases.

Archive

This is the archive manager. It is responsible for taking live session data from Redis and saving this to the PostgreSQL database. It is also responsible for supplying data from the database to the reporting API for retrieval by the client or LumenVox administration or deployment portals.

Asr

This is the ASR DNN engine. It is linked with the session manager, and it processes all ASR & Transcription transactions. Results are placed in Redis for retrieval. It receives messages from RabbitMQ on what audio needs to be processed. The latest version ships with our new DNN ASR engine providing better performance and accuracy. The new engine also offers enhanced features like aliases, dialect specific processing, improved scoring algorithm, enhanced transcription, phrase lists and weighting. The new engine is much faster utilizing less resources than its predecessor thus making scaling more manageable. Note that due to the new scoring mechanism, clients are advised to review any existing thresholds set.

Audit

This service connects to audit database in PostgreSQL. This is used by the voice biometric services only. It is used to audit modifications to configurations the configuration, deployment, and biometric identity tables.

Binary Storage

Connects to MongoDB and reads/writes from/to the MongoDB. It is also responsible for the encryption/decryption of data stored within Mongo. This includes audio files and grammars.

Biometric-active

This is the Voice Biometric active DNN engine and is responsible for all active voice biometrics processing receiving requests from the biometric APIs

Biometric-api

This is the REST interface for active voice biometrics. It provides APIs for enrollment, verification, and identity management. This service talks to the deployment, configuration, binary service, active verifier and to lumenvox API services (for text validation).

Biometric-identity

Responsible for managing the identities for the active engine for voice biometrics. It is also responsible for maintaining the links between identities and enrollments.

Configuration

Responsible for managing the configuration for various deployments for voice biometrics only. It receives input via JSON. And manages the retrieval of enrollment and verification configurations from the PostgreSQL database.

Deployment

This service manages deployments within the cluster. For each deployment, it maintains the list of configurations and connection strings along with any other data specific to a deployment. It also handles encryption and encryption keys along with key rotation. The service is also used to manage the deployment e.g., create a new deployment Id, and export/import deployment information.

The cluster master key gets created when the deployment service starts for the first time and is stored in the database. The cluster master key is encrypted with the cluster GUID. The customer keys are then created for each deployment. These are encrypted using the cluster master key.

© 2024 LumenVox

Deployment-portal

This service manages the web portal for the tenant admins. It facilitates the tenants to access their own deployment. The service supports the customization of configurations and manages the diagnostic tests.

Grammar manager

The new Grammar management service co-ordinates the grammar compilation, caching, parsing & DTMF processing for ASR & Transcription transactions. It is also instrumental in CPA & AMD transactions; grammar storage, retrieval & housekeeping and handling of global grammars and phrase lists.

ITN

This service manages all text normalization functions including inverse text normalization, punctuation & capitalization and redaction on Transcription interactions.

License

This service manages the licensing of the deployments. It validates each deployment against the LumenVox licensing server. The licensing service is responsible for making a deployment as valid, if not, the various APIs will not function.
It also collects counters for transactions for all products and communicates this to the centralized LumenVox licensing service for billing and product usage monitoring.

LumenVox-api

This service supports the Speech APIs made available to clients to process transactions for speech products including ASR, TTS, CPA and AMD using gRPC.

Management-api

This facilitates the use of the REST management APIs and is a wrapper around deployment and configuration services which allows customers to manage these.

Persistent-volume-directory-setup

This service is used to create the connection to the persistent volume. It is not a service that runs continuously. It is used to update resources e.g., DNN language models in the persistent storage.

Reporting

This service is used for voice biometrics to generate the reports and make the extracted data available to the reporting APIs

© 2024 LumenVox

Reporting-api

This facilitates the speech reporting APIs using gRPC and is used for the Analysis portal.

Reporting-bio-api

This REST interface is used to connect to the reporting services and expose reports to the client for voice biometrics.

Resource

This service downloads the public language packs to the cluster for ASR, TTS, and voice biometrics. It stores the languages packs into the shared volume for the cluster and makes resources available to each service as needed.

Session

Currently used for speech products only. It manages the sessions and interactions and acts as a dispatcher between the LumenVox API service and other services.

Transaction

This service is responsible for writing voice biometric transactions into the database. Voice biometric sessions are stored in Redis until the session is completed, when the transaction service receives a message to retrieve data and write the session and transaction data to the PostgreSQL database. This service is also used for operations reporting where each API call to the biometric interface is logged as an operation and stored for audit purposes.

Tts

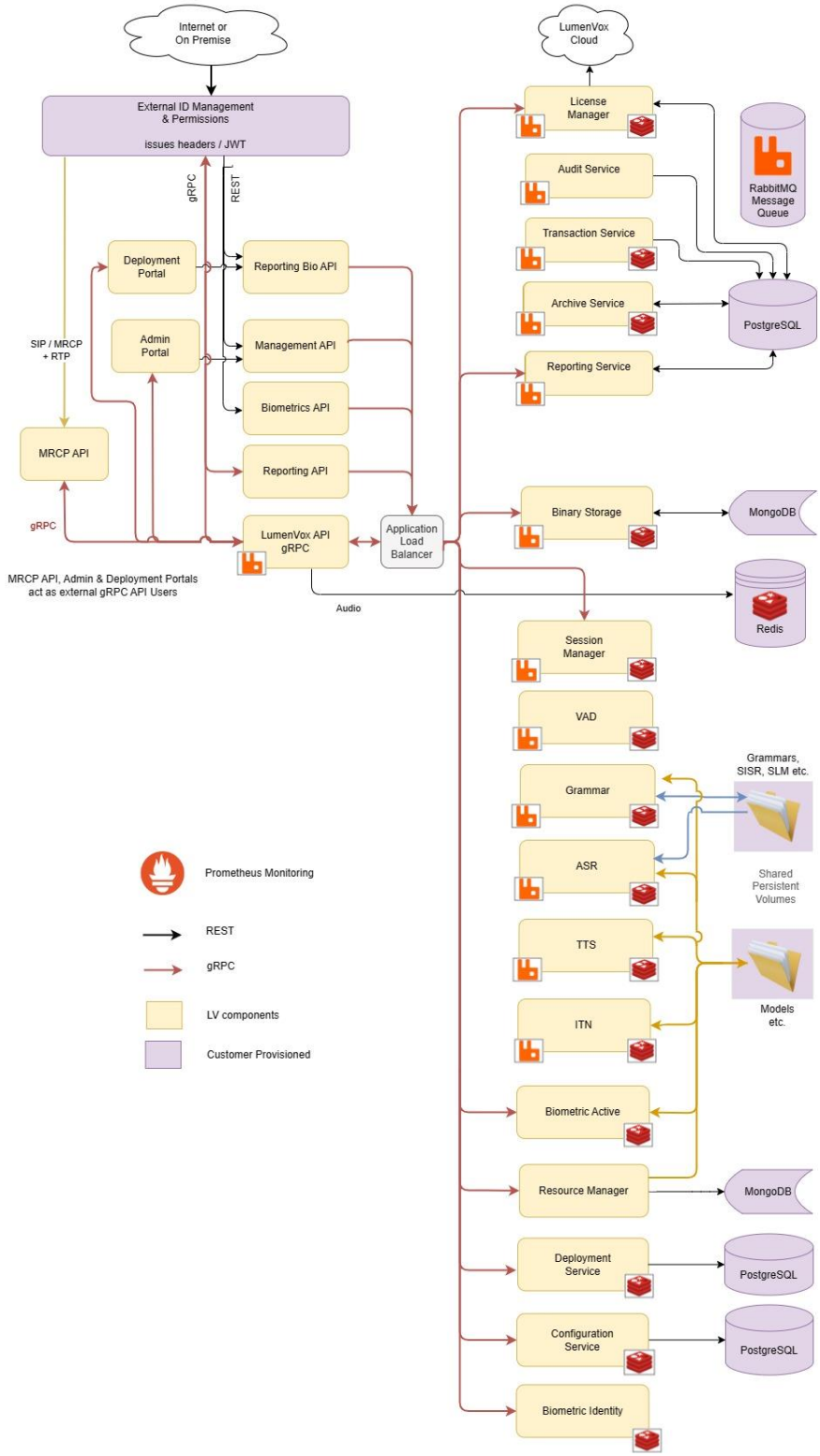
This is used for the TTS engine.

Vad

This is the media service that manages the VAD algorithm and assists with transcription audio streaming.

© 2024 LumenVox

LumenVox Microservices Diagram *- the following is an architecture diagram that Includes all services.*



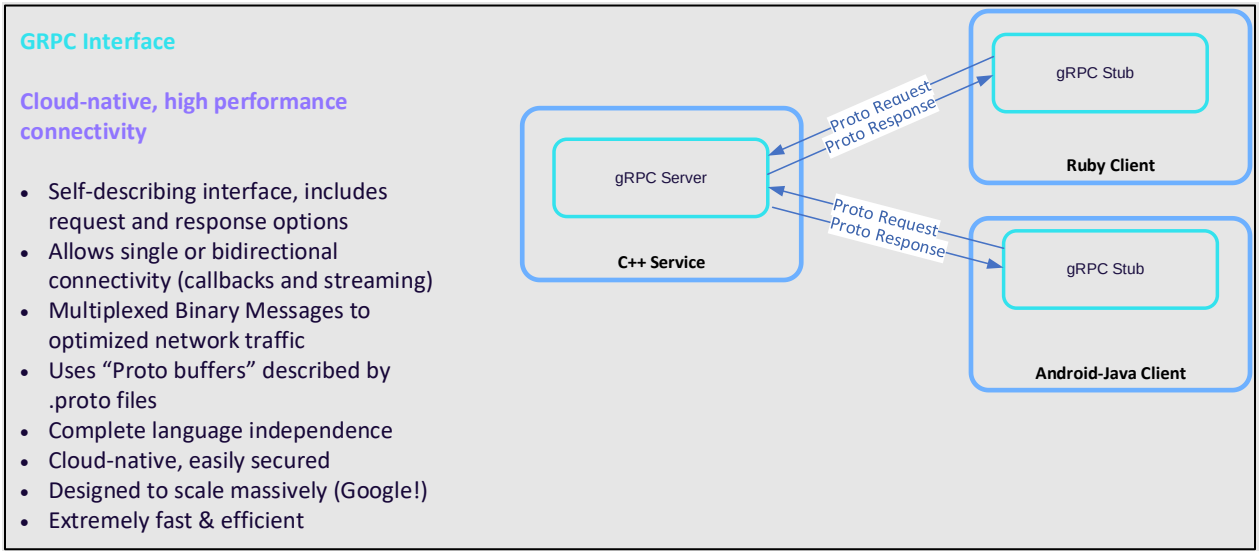
Communication Protocols

Supported Standards to Access LumenVox Services

gRPC

gRPC is a cross-platform open-source high performance Remote Procedure Call (RPC) framework, initially created by Google. This is the preferred and recommended protocol for communication with LumenVox software. gRPC can run in any environment and can efficiently connect services in and across data centers with support for load balancing, tracing, health checking and authentication. It is optimized for scalability and is cloud and microservices native. All major programing languages can access our API via gRPC. Supported programing languages can be located here [Supported languages | gRPC].

The service proto files can be obtained from here. The client should access the relevant release version [Protocol Documentation (lumenvox.com)]



MRCP

Media Resource Control Protocol is a communication protocol between applications (voice IVR platforms) and the ASR and TTS resources serving them (taking in the audio or outputting the text). Primarily used by voice application platforms, call centers, call recording providers, telephony trunks, and network switches, using the W3C standards.

A subset of the standard is specific for speech recognition and voice biometrics. MRCP uses 'methods' to control speech resources: to start or stop a recognizer, set parameters, load grammars, or control a speech synthesizer (for TTS). MRCP uses web technologies to deliver information and commands.

- MRCPv1 used RTSP (real time streaming protocol) where session management and the MRCP content are sent together.
- MRCPv2 uses SIP (session initiation protocol) where a session is established first, then MRCP content is sent separately, allowing both ASR and TTS in one session.

Use of MRCPv2 is prevailing in customer service platforms. We support all major voice platforms and IVRs using MRCP. The MRPC services need to be run on their own virtual machine using Docker, as MRCP uses UDP and TCP ports which are currently not supported by Kubernetes. To support this, a media server that sits outside the Kubernetes cluster for communicating with MRCP clients is used. The client app communicates via MRCP to our media server, the MRCP API, which in turn uses gRPC to communicate with the LumenVox API.

This does limit the ability for MRCP to scale well. However, this approach allows massive connectivity: many MRCP API servers can connect to a single LumenVox Kubernetes Cluster.

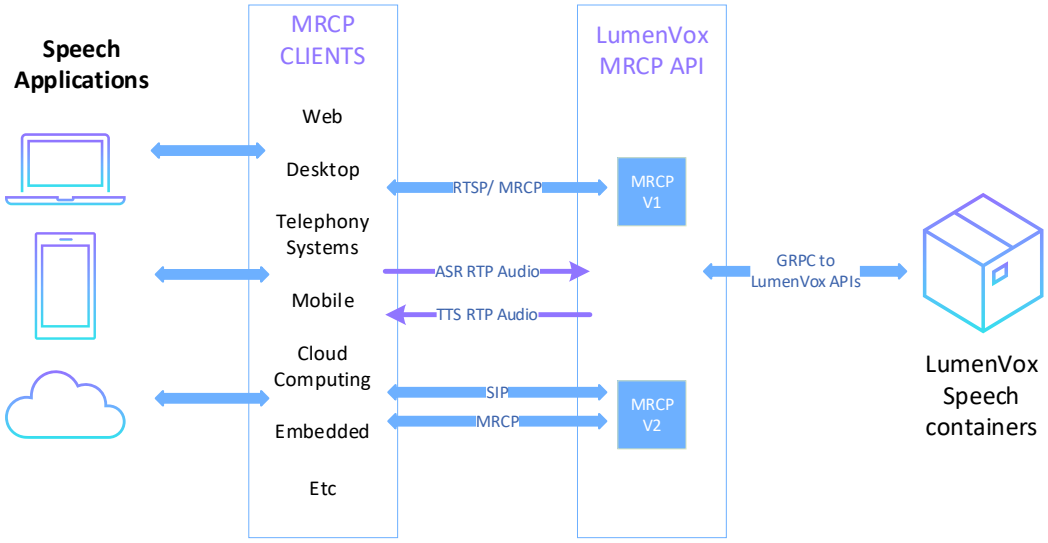
The following links provide further information:

- How to connect with MRCP Version 1 and MRCP Version 2
- MRCP Connectivity – LumenVox Knowledgebase articles

Refer to Appendix 7 for MRCP API server installation steps.

If clients want to just install a simple MRCP installation to test the MRCP API, they could make use of our Simple MRCP client. More information can be obtained here: https://www.lumenvox.com/knowledgebase/index.php?/article/AA-01633/13/

MRCP services are deployment-specific and there is no option to change the deployment ID used by the service. If customers want to provide different tenants (deployments) access to MRCP, each will require their own MRCP service configured for their own deploymentId.



REST

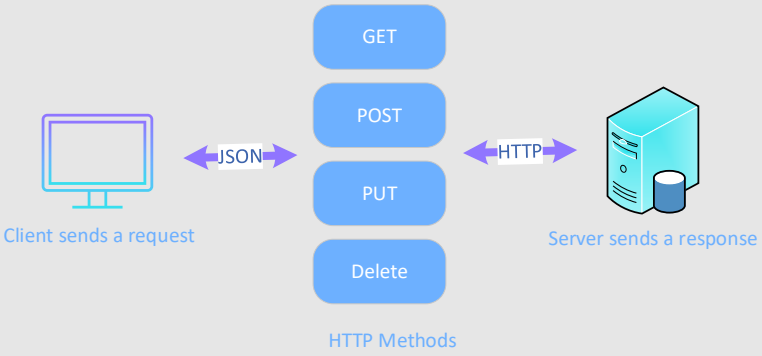
A subset of LumenVox clients use REST APIs with HTTP or Web Sockets for their communications protocols. REST / HTTP is a traditional model for communications with backend servers. It relies on a client (web application) initiating a request, such as obtaining data from a database, and is stateless. This model doesn't work well for applications with real time data access requirements. The REST interface is used for the Voice Biometric APIs, Voice biometric reporting APIs, and management APIs.

The REST API guide can be obtained from here. The client should access the relevant release version [LumenVox API Documentation]

RESTful Interface

Widely used internet connectivity

- Easy to implement & test, simplistic interface mechanism
- Operates on existing browser-style connections (no fancy tools needed)
- Universally supported. Can use command line (curl) tools
- Limited streaming support
- Limited callback handling
- Doesn't scale well



Visit the LumenVox Knowledgebase for technical information on the supported communication protocols

Administration & Maintenance Tools

Admin Portal & Deployment Portal

With the cloud-native technology stack come the features required to maintain an installation of our software via a unified web portal UI. Partners and clients can manage the implementation using their own interfaces or use our portal to call various APIs and perform the required functions. The web-based portal can be used with any public / private cloud or on-premises implementation.

The portal currently serves two user communities:

1. Cluster administrators: at a large implementation of platform providers, the cluster admin manages deploying the software to the tenants.
2. Tenant administrators: administrators at the individual tenant (or administrators at small or on-premises installations) manage their own deployment.

A 'Deployment' is a tenant-level instance. It has a unique Deployment ID, for which the tenant gets licensed. To consume any LumenVox product a Deployment must be created with a Deployment ID, and this can be instantly accomplished via the portal.

For reasons of security and segregation of duties between the two types of users, we offer two interfaces:

1. **Admin Portal** – for the cluster admin, who sets up each licensed tenant with their connection strings, encryption, and passwords to services and databases.
2. **Deployment Portal** – for the tenant admins, who can only access their own deployment. Within it they can customize the default configuration by tweaking parameters and options, verify that all services are up and running, and run diagnostics if there's a problem.

The Admin Portal has an access point into the Deployment Portal, via a Deployment Portal button, available in Deployment Details page.
In some implementations, the cluster admin might be the one performing administration for the deployments. In other implementations it will be more appropriate for the cluster admin to have no such access. We make it easy for IT to control this by firewall rules, white-listing access by using the host site name format: ingress_host_name.client_hostname_suffix
For example:

1. admin-portal.testmachine.com
2. deployment-portal.testmachine.com

Customers can choose the ports that are exposed externally. The two portals are exposed on separate (configurable) ports to aid with firewall and access configuration.

© 2024 LumenVox

If using self-signed certificates, then the following steps must be taken should you wish to view the portal from your desktop. Please see instructions in Appendix 9.

Analysis Portal

Our legacy product line offered a desktop app called Speech Tuner, that could be used to analyze the performance of your speech application and tune it.

The Analysis Portal replaces that desktop tool with an easy-to-use web interface, integrated with the Deployment Portal; it is intended for tenant-level only access, and analysis of speech performance within the deployment. The initial interface allows deployment administrators to:

- Gather data - create an analysis set of speech transactions (audio and its resulting transcription)
- Transcribe the data manually
- Get accuracy specifics by comparison of the manual annotation to the Engine output, identify problems, and if a grammar was used – in which grammar.
- Get accuracy statistics and reports

The tool is available for ASR, CPA, AMD & Transcription transactions. Further features will be made available in future releases.

Kubernetes Installation Steps

The following information provides high-level steps to be followed to install the LumenVox software within a Kubernetes environment. Appendix 1 (Google GKE) and Appendix 2 (Amazon EKS) provide further information and steps for these two cloud providers. Further providers may be added in due course. LumenVox offers a "Quick start" option for on-premises installations making use of Kubeadm which automates most of the steps to be followed below. See Appendix 3 for further steps.

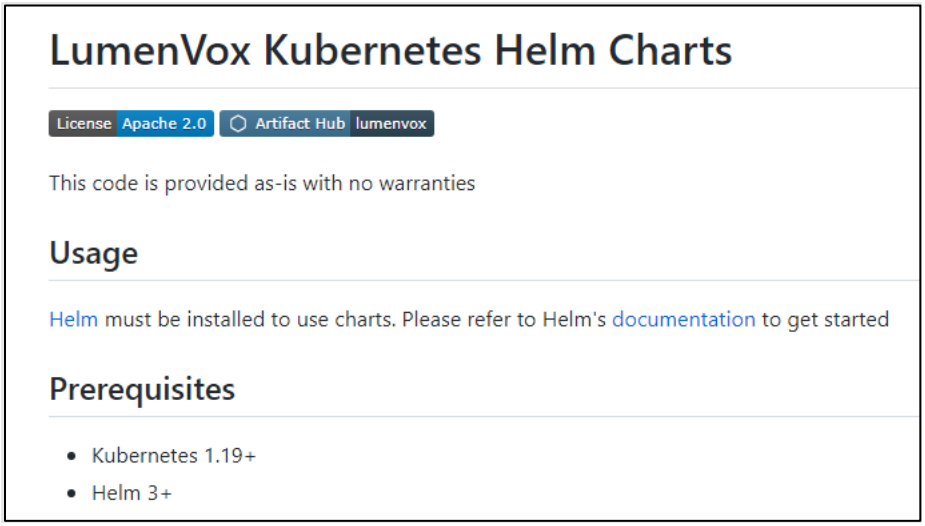
LumenVox recommends that clients set up a Kubernetes cluster environment that is not reliant on external services for MongoDB, PostgreSQL, Rabit and MongoDB and the client can soon begin testing using the various APIs. Once the setup is successful then the Kubernetes environment could be set up using the external services. Note that when setting up the connection strings to these external services, the following characters are not supported within a password: **[ ] { } ( ) , ; ? * = ! @ |**

LumenVox is able to supply the values files used for the Helm commands. Examples of these can be found here: [GitHub - lumenvox/helm-charts: LumenVox Kubernetes Helm Charts](). We would just require the IP address details for the persistent storage device along with the usernames and IP addresses for the various prerequisite components like Redis and

MongoDB. The client needs to populate the secrets file template with their base64 encoded passwords. The template can be obtained from Appendix 6.

Steps

1) Create a Kubernetes cluster
    a. This will host your Kubernetes nodes
    b. IP created can be noted for association with DNS & API calls
    c. Cluster name and networking information (e.g. publicly available) to be set up
    d. Version 1.26.4 is currently recommended – see latest minimum requirements in GitHub - lumenvox/helm-charts: LumenVox Kubernetes Helm Charts



2) Create the node pools
    a. This will host your various pods
    b. Numbers to be created will depend on call volumes and products used
3) Configure the nodes
    a. This should include specifying the operating system
4) Set up the node security
5) Set up node networking

6) Install KubeCTL or equivalent version e.g., Google CLI which installs KubeCTL as part of the installation package. This used to connect to the Kubernetes cluster from your local machine.

7) Connect to the cluster and configure KubeCTL in order for KubeCTL to communicate to the new cluster and submit Kubernetes commands.

8) Install Linkerd

    a. This is a service mesh that LumenVox recommends for effective interaction management & load balancing between the various pods.

9) Install Jaeger

    a. This is a plugin for Linkerd. It is a monitoring tool that keeps track of latency, communication success between pods and bottlenecks. It can monitor how much time is spent for transactions within the various pods.

10) Install Linkerd Dashboard

    a. This is a visual dashboard used to monitor and display latency and uses Jaeger behind the scenes. It also works as a health check for the service mesh.

11) Install Helm

    a. This will be needed to run the various helm charts

12) Use Helm to install Nginx Ingress .

    a. NGINX is a load balance into the Kubernetes cluster and works as a reverse proxy.

13) Create namespace and change to *lumenvox*

    a. A namespace is required to install the pods into. A namespace can have multiple tenants (deployment IDs). At this stage LumenVox does not support multiple namespaces.

    b. Commands are run to activate and make *lumenvox* namespace the default

14) Set up TLS for ingress

    a. Used to set up TLS and apply a self-signed certificate or client alternative

15) Apply secrets file

© 2024 LumenVox

a. The secrets file must be populated with the various component passwords stored as a base64 sting. The following example is of the secrets.yaml file

```yaml
# This file should be applied to your Kubernetes cluster before
# installing any LumenVox Helm Chart. This assigns credentials
# for various services. You can change these to meet your specific
# installation requirements.

# Be sure to adjust namespace values if needed too

apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: rabbitmq-existing-secret
type: Opaque
data:
  rabbitmq-password: examplefWnc22423fsdfs2svdvWA==
---
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: postgres-existing-secret
type: Opaque
data:
  postgresql-password: examplefWnc22423fsdfs2svdvWA==
  postgresql-postgres-password: examplefWnc22423fsdfs2svdvWA==
---
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: redis-existing-secret
type: Opaque
data:
  redis-password: examplefWnc22423fsdfs2svdvWA==
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: mongodb-existing-secret
type: Opaque
data:
  mongodb-root-password: examplefWnc22423fsdfs2svdvWA==
---
```

16) Run installation of the helm cart

a. The various helm charts required can be located at https://github.com/lumenvox/helm-charts

b. The resources file can be used to override any default settings

17) Run LumenVox sample scripts

a. Sample speech python scripts are available for clients to easily test out the LumenVox API and verify that everything is installed correctly. This can be found here: LumenVox · GitHub

Troubleshooting

Should your containers not fully deploy then the following steps can be taken to troubleshoot the problem:

- Are all the pods running
- Are there any errors in the pods not running
- What are the errors?
- Check deployment pod status and log files. This is the most important pod to getting other pods up and running

Potential Installation pain-points

The following list provides some examples of potential areas that could affect your installation for troubleshooting purposes:

- Network between components e.g. between Kubernetes and Mongo DB, Redis, RabbitMQ, Postgres & the persistent volume.
- Using different clusters and the communication with them
- The way clusters are set up in different host providers can differ e.g. between Alibaba, Azure, Google, Amazon...
- Connectivity and configuration to MongoDB, PostgreSQL, RabbitMQ, Redis and the persistent storage can differ per client installation. The version or type of these five components can also differ
- Passwords for various components contains unsupported characters
- TLS connectivity
- Connectivity to licensing service & LumenVox resources

© 2024 LumenVox

Logging

Installation steps should be taken to install the logging framework of your choice.

Monitoring

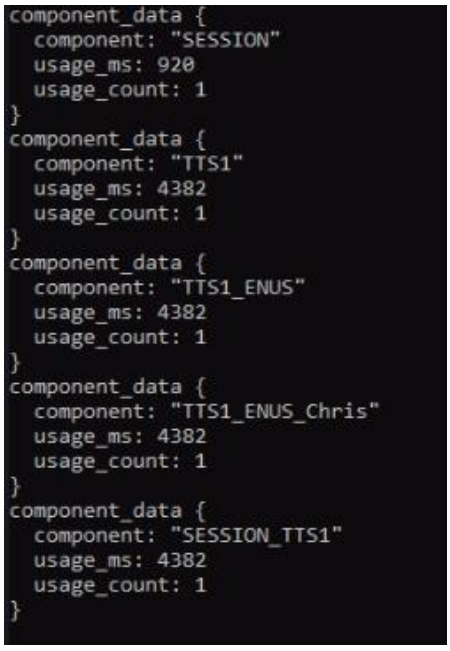
Installation steps should be taken to install the monitoring framework of your choice.

Licensing

LumenVox requires all installations to connect to the LumenVox licensing service to validate the license and report back on product usage. The client is required to open firewalls to communicate with https://licensing.lumenvox.com via port 443. The LumenVox software supplies information from the cluster and receives a signed response validating the license.

Clients can obtain a copy of the license report by calling Reporting.UsageReportRequest – the gRPC API. The API will return usage stats based on the component supplied as input. If no component is submitted, then stats for all components utilized will be returned.

The following is an example of the output.



Resources

Resources for the various speech and voice models need to be acquired from LumenVox's resource repository. Clients are required to open firewalls so that the containers can obtain the relevant resources stipulated in the helm charts. Access to the following link is required: https://lumenvox-public-assets.s3.amazonaws.com/model-files/*

© 2024 LumenVox

APPENDIX 1 – Setup in Google GCP (GKE Creation)

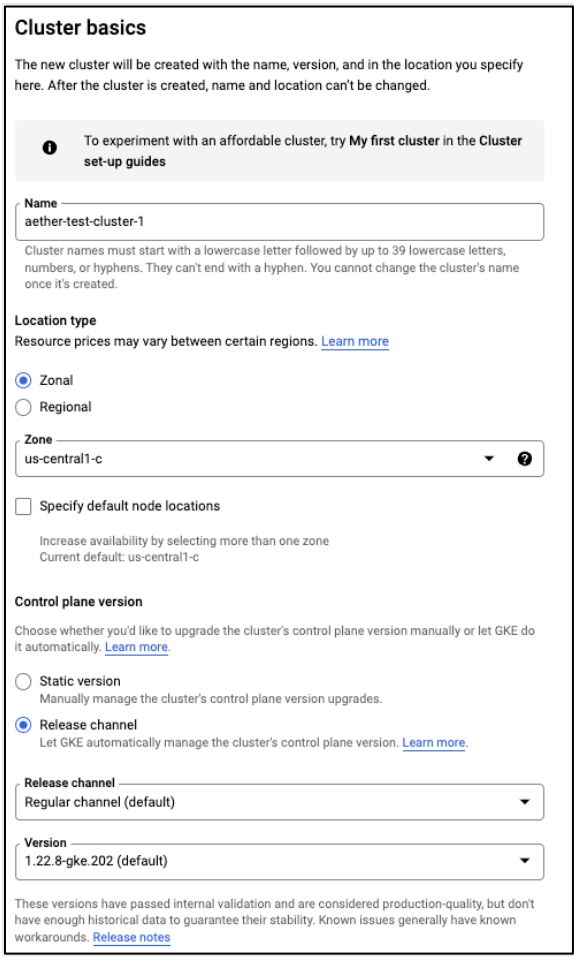
The following steps can be followed to install the LumenVox software within Google's Kubernetes platform.

Create Kubernetes Cluster

Note: DNS associated with cluster can be noted for API calls into the LumenVox software

Set up cluster basics

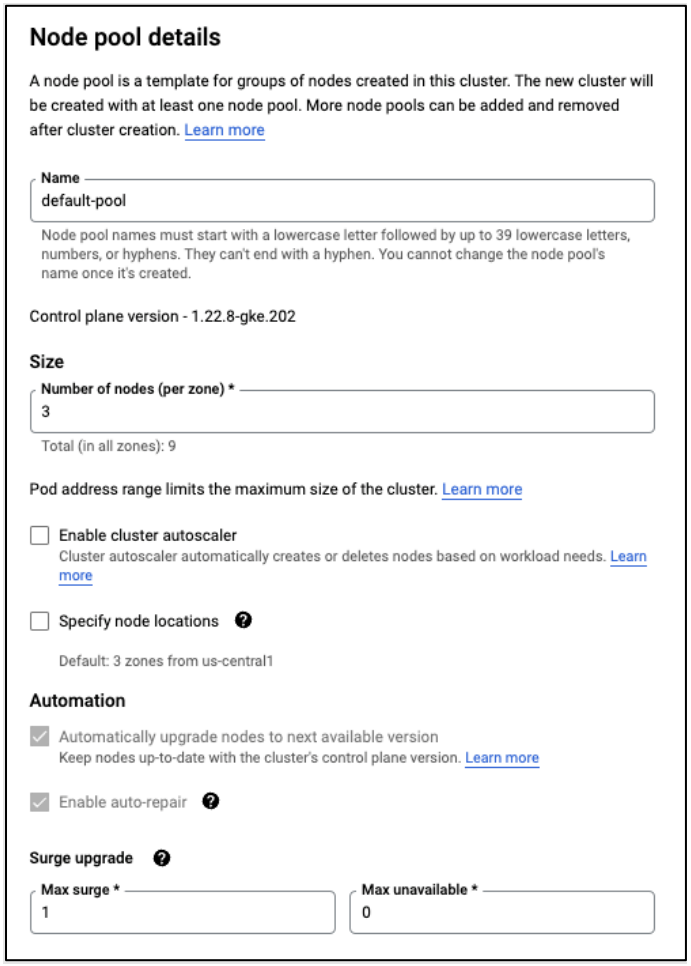
- Provide a cluster name and networking information (e.g. publicly available) to be set up
- Version 1.26.4 is currently recommended – see latest minimum requirements in [GitHub - lumenvox/helm-charts: LumenVox Kubernetes Helm Charts]



Set up Node pool details

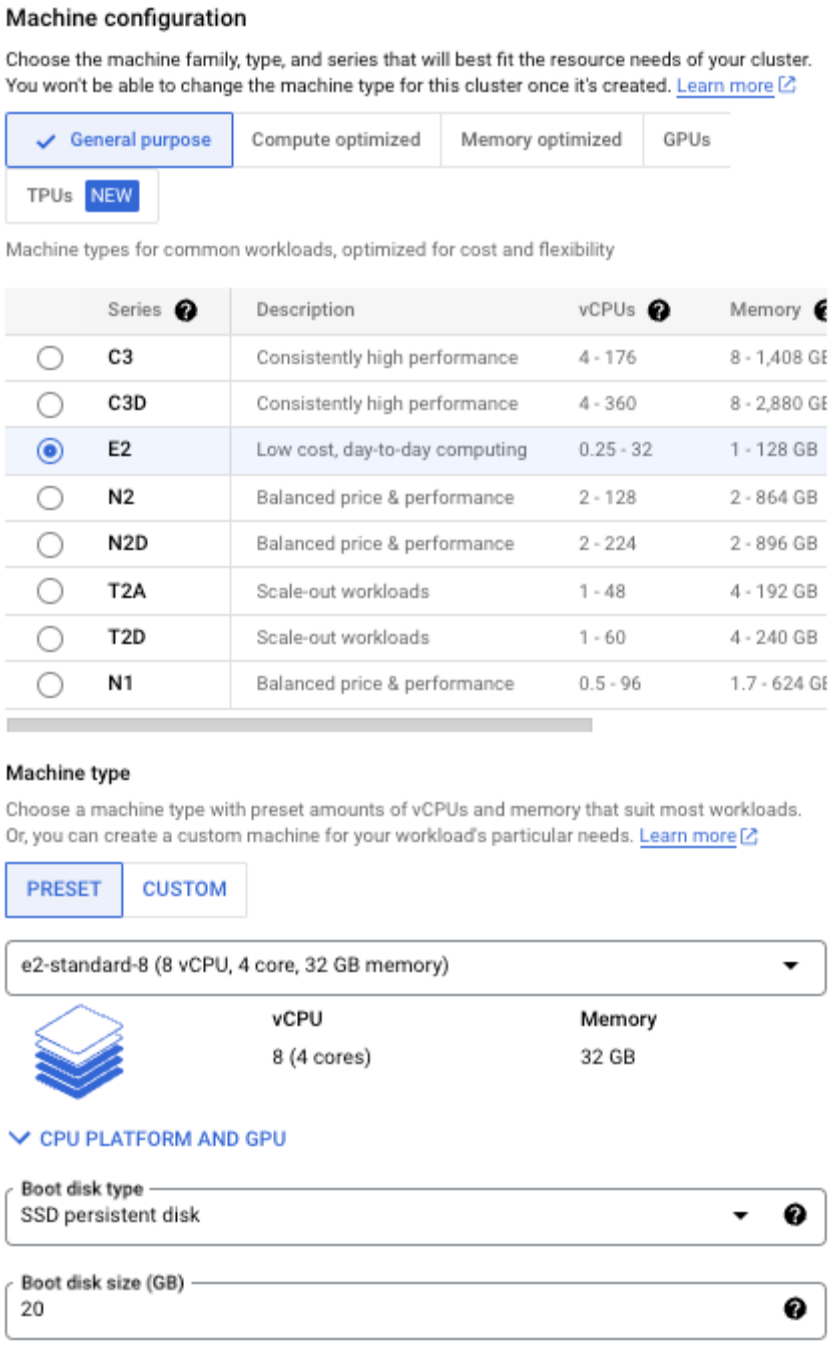
- This will host your various pods
- Numbers to be created will depend on call volumes and products used

The following default values are recommended but should be changed as per the client's individual requirements:



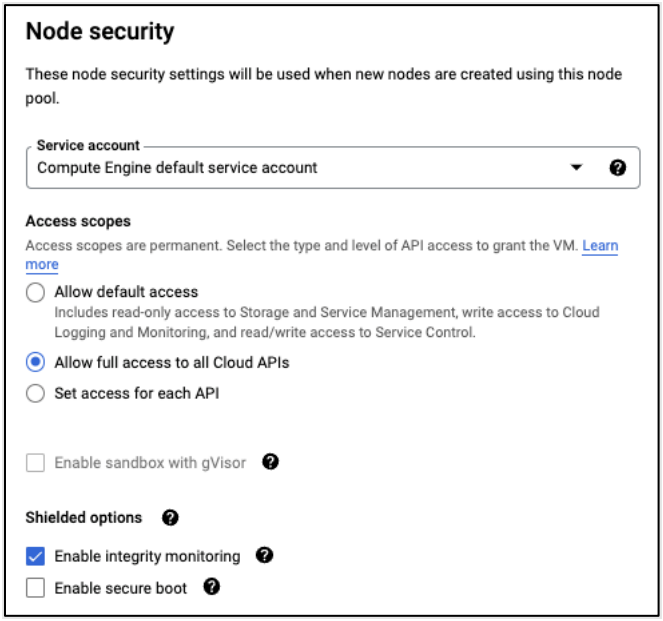
Set up node configurations

The following default values are recommended however the client should select the operating system that best suits their needs. LumenVox recommend 8 vCPUs, 32GB memory per node.



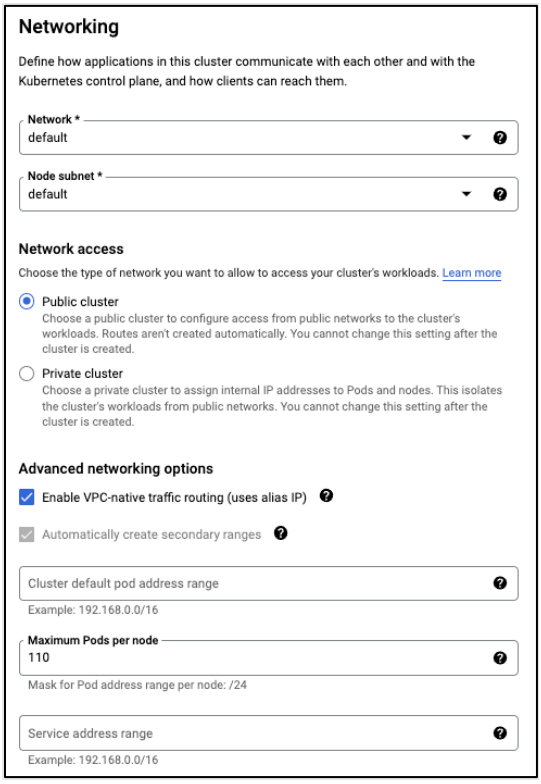
Set up node security

The following default values are recommended but should be changed as per the client's individual requirements:



Set up node networking

The following default values are recommended but should be changed as per the client's individual requirement. LumenVox recommends using a public cluster network access.

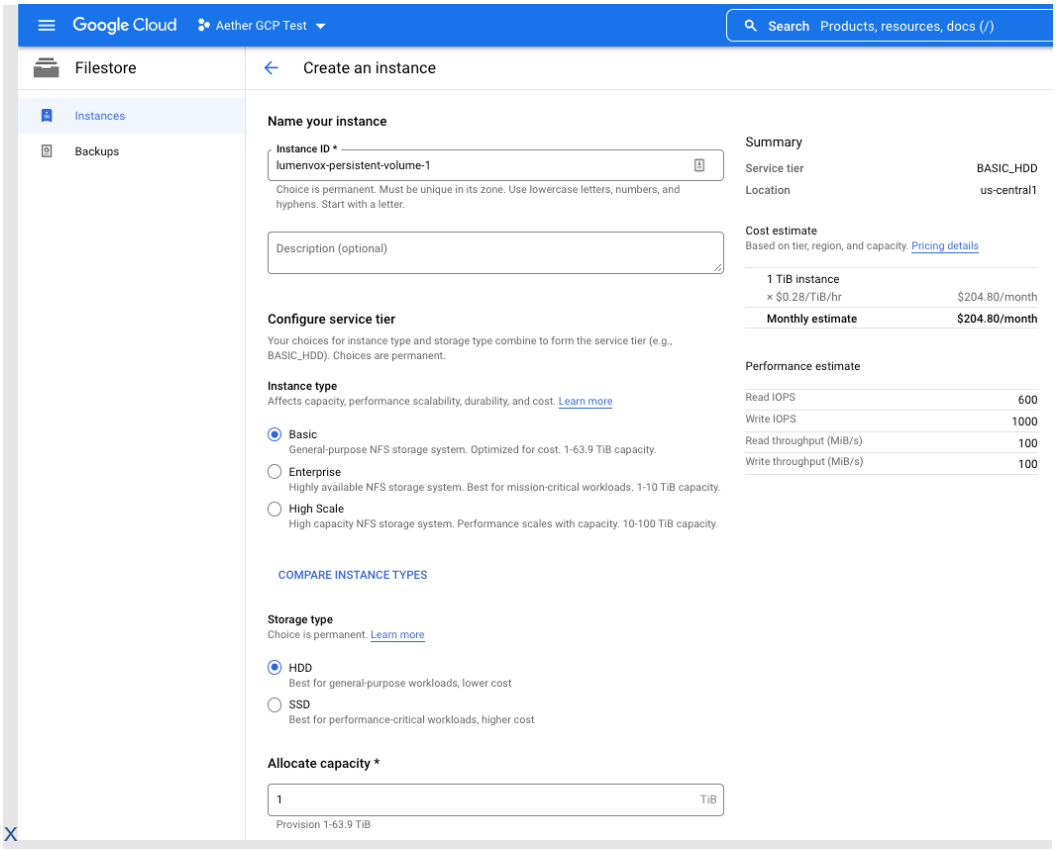


Create VPC Firewall Rule for Rabbit MQ

- Google doesn't add automatically add this

Add persistent storage

The following default values are recommended but should be changed as per the client's individual requirements:



Install GCloud CLI

This will include the required KubeCTL

Download from: https://cloud.google.com/sdk/docs/install

Connect to Cluster (configuring KUBECTL to talk to new cluster) ---this is on the user machine – e.g. remote connection tool.

```
1. gcloud container clusters get-credentials aether-test-cluster-1 --zone us-central1-c --project
aether-gcp-test
```

Install Linkerd CLI

This is run on KubeCTL and will install to local machine

```
1. curl --proto '=https' --tlsv1.2 -sSfL https://run.linkerd.io/install | sh
```

Add the Linkerd CLI to your path

```
1. export PATH=$PATH:$HOME/.linkerd2/bin
```

Install Linkerd

```
1. linkerd check --pre
2. linkerd install --crds | kubectl apply -f -
3. linkerd install | kubectl apply -f -
4. linkerd check
```

Install Jaeger

```
1. linkerd jaeger install | kubectl apply -f -
```

Install Linkerd Dashboard

```
1. linkerd viz install | kubectl apply -f -
```

Should be ever need to uninstall Linkerd for troubleshooting purposes (e.g. deleting a cluster or you have issues with the Linkerd side cars you can use the following commands:

Uninstall Linkerd

```
1. linkerd viz uninstall | kubectl delete -f -
2. linkerd jaeger uninstall | kubectl delete -f -
3. linkerd uninstall | kubectl delete -f -
```

Install Helm

```
1. curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
2. chmod 700 get_helm.sh
3. ./get_helm.sh4.
```

© 2024 LumenVox

Install nginx Ingress

```
1. helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
2. helm repo update
3. kubectl create ns ingress-nginx
4. helm upgrade --install ingress-nginx ingress-nginx --repo https://kubernetes.github.io/ingress-nginx -n ingress-nginx --create-namespace --set controller.hostNetwork=true --set controller.allowSnippetAnnotations=true
```

Create LumenVox Namespace

```
1. kubectl create namespace lumenvox
```

Change NameSpace

```
1. kubectl config set-context --current --namespace=lumenvox
```

Setup TLS for Ingress

The example below shows guideline values, clients can amend as needed e.g. certificate validity period and subject alternative name.

```
1. openssl genrsa -out server.key 2048
2. openssl req -new -x509 -sha256 -key server.key -out server.crt -days 3650 -addext "subjectAltName = DNS:lumenvox-api.testmachine.com, DNS:biometric-api.testmachine.com, DNS:management-api.testmachine.com, DNS:reporting-api.testmachine.com, DNS:admin-portal.testmachine.com, DNS:deployment-portal.testmachine.com"
3. kubectl create secret tls speech-tls-secret --key server.key --cert server.crt
```

Apply Secrets File

```
1. kubectl apply -f lumenvox-secrets.yaml
```

Installation of Helm Charts

Deploy Public Chart

```
1. helm install lumenvox lumenvox/lumenvox -f speech-values.yaml -n lumenvox
```

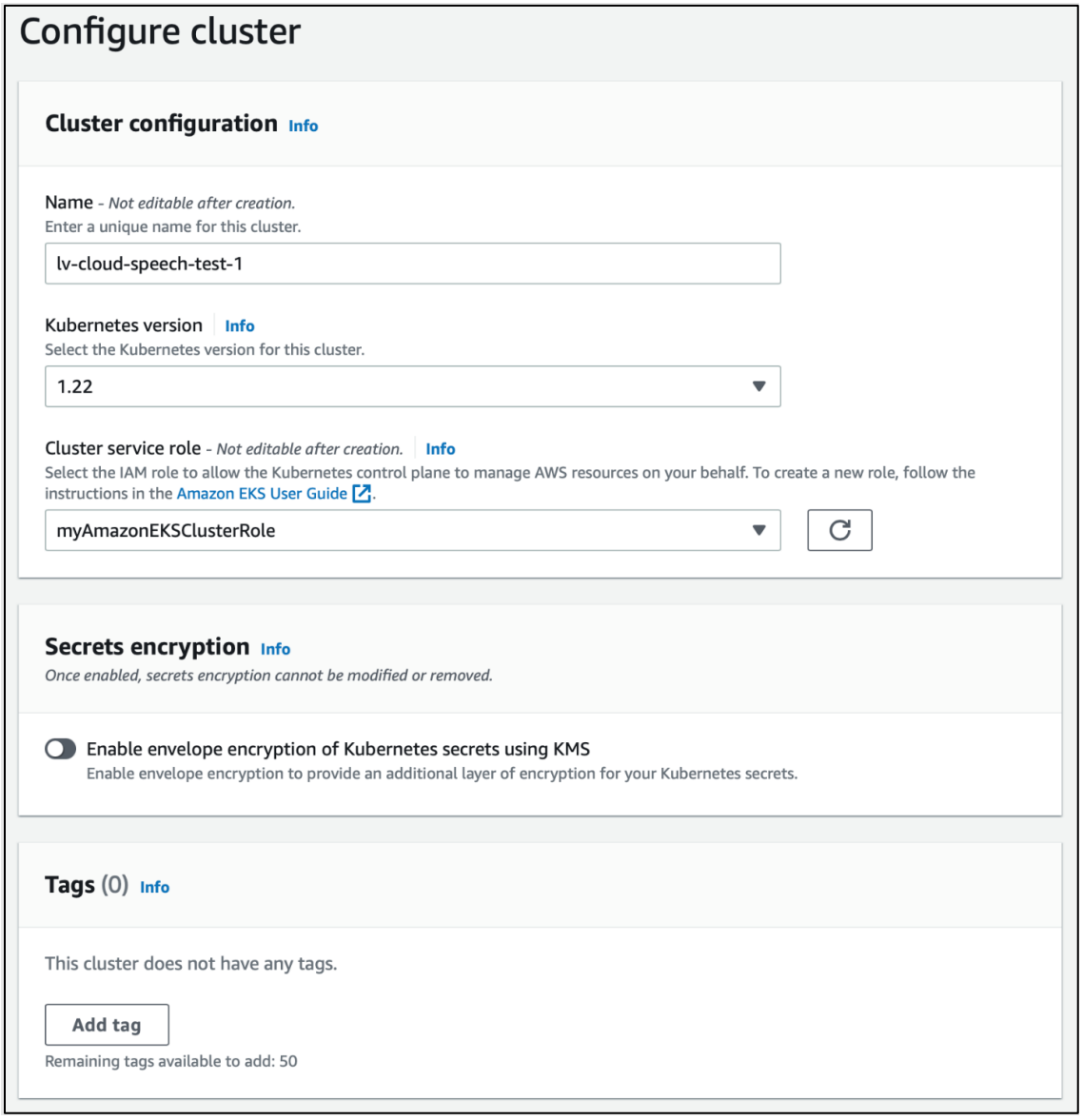
See Appendix 4 for final steps to for complete installation by setting up a deployment

APPENDIX 2 – Amazon Kubernetes Service (EKS)

The following steps can be followed to set up the LumenVox software within Amazon.

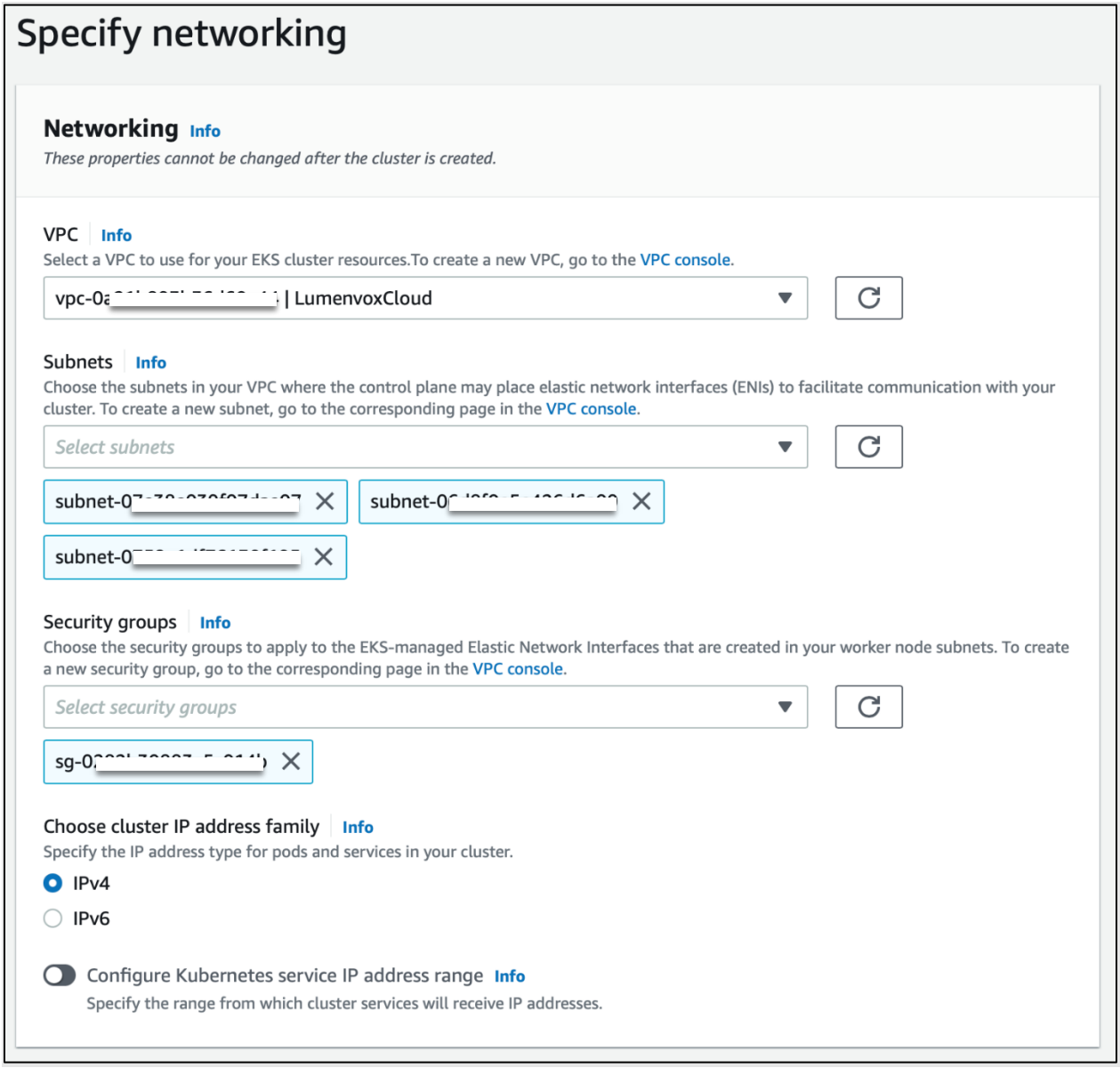
Create EKS Cluster

- Provide a cluster name and networking information (e.g. publicly available) to be set up
- Version **1.26.4** is currently recommended is currently recommended – see latest minimum requirements in [GitHub - lumenvox/helm-charts: LumenVox Kubernetes Helm Charts]



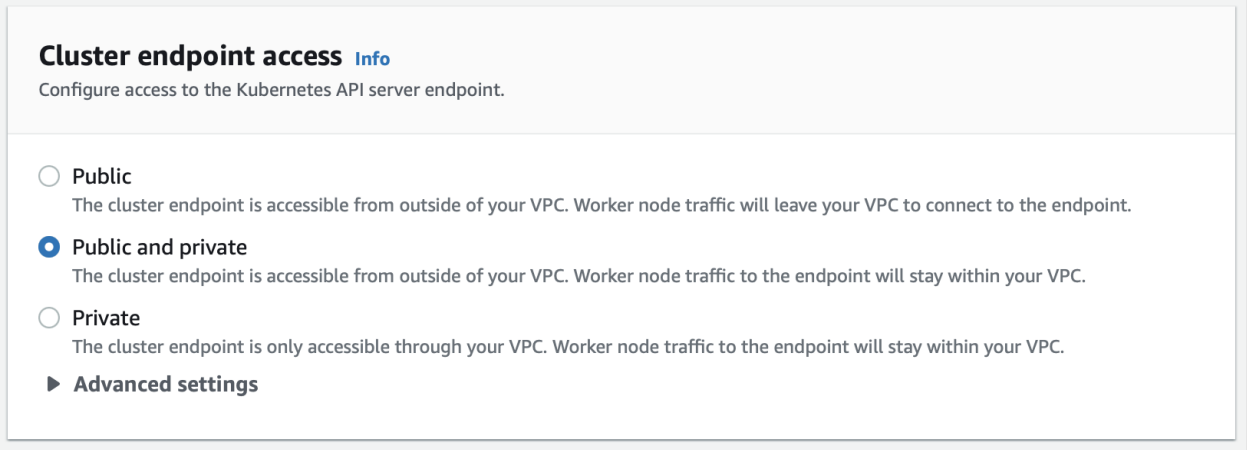
EKS Cluster Networking

- Specify VPC, select subnets and add the relevant security groups (we used the default VPC security group)
- VPC to be created beforehand if it doesn't exist.



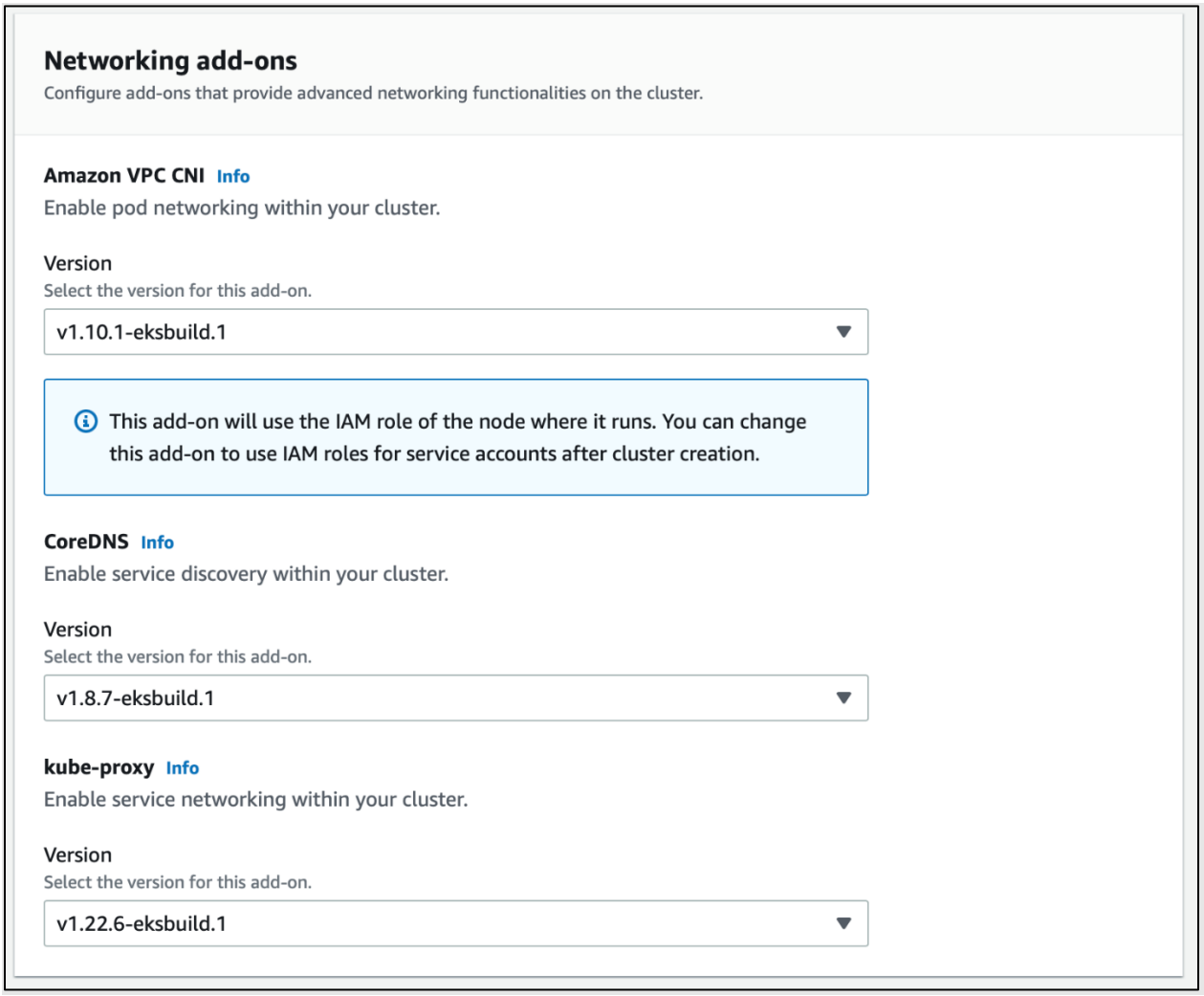
Select Cluster Endpoint Access

We recommend the following settings:



Networking add-ons

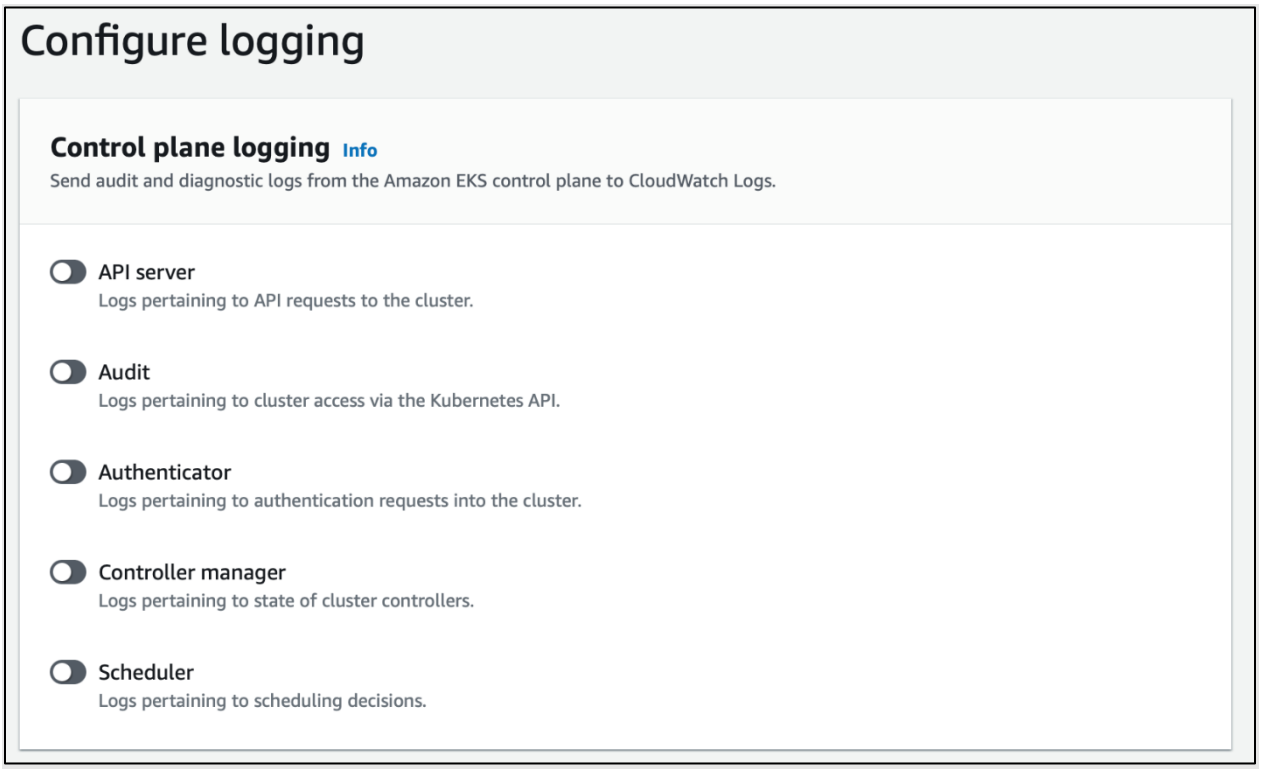
We recommend the following settings:



Configure logging

We recommend the following settings:

.



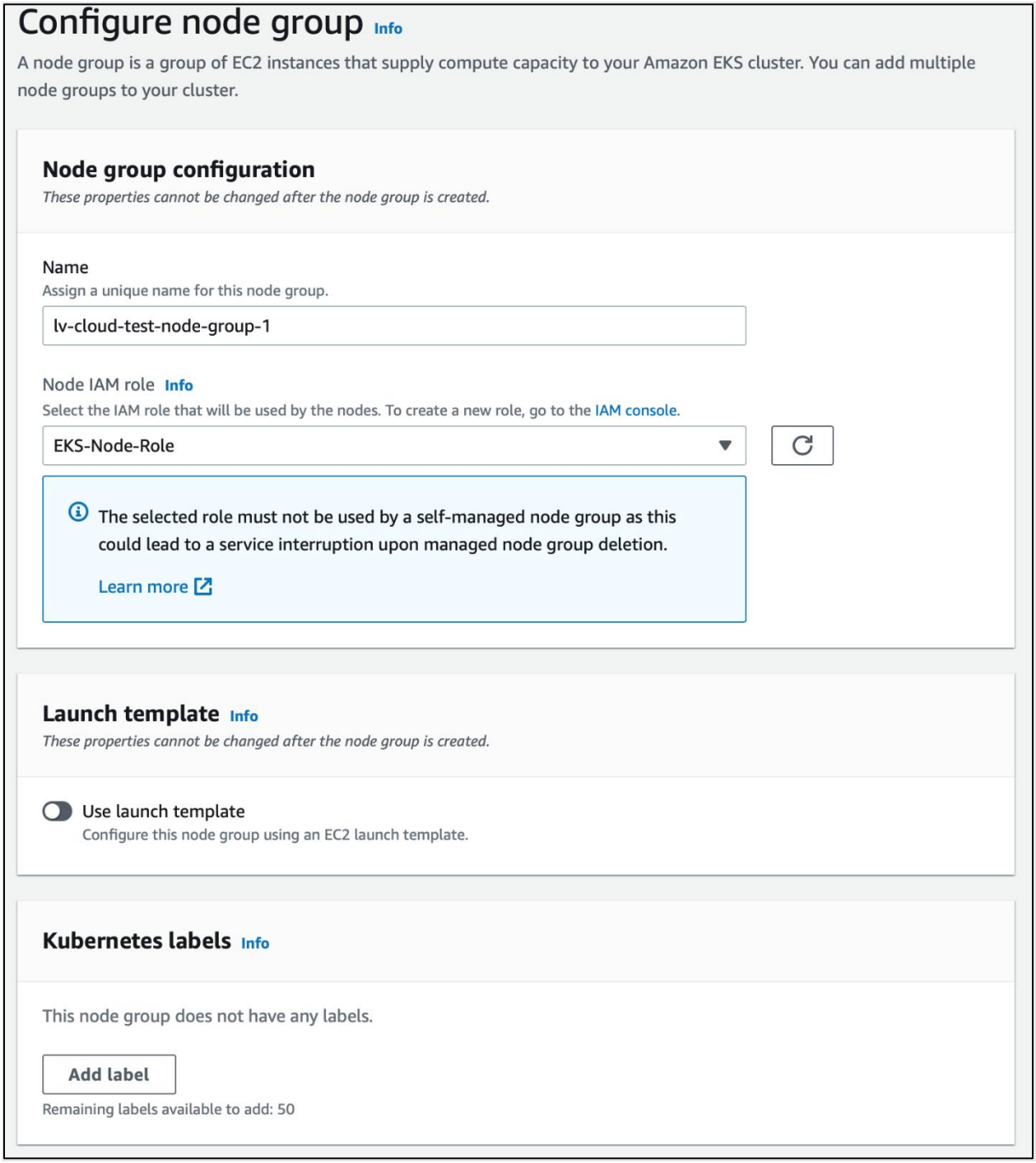
- Review and click on create
- Cluster creation can take several minutes to complete. Monitor for any errors.

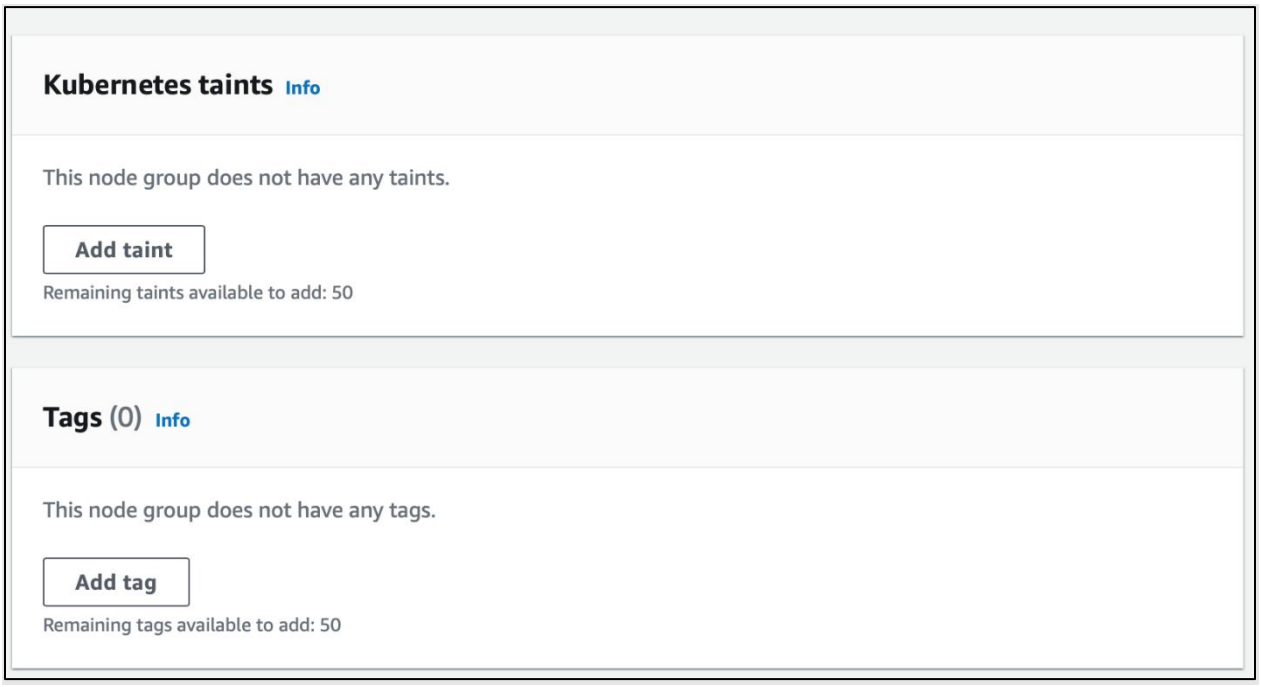
Create node group and create node

- Click on Compute and then Add Node Group

Configure Node Group

The following default values are recommended but should be changed as per the client's individual requirements:

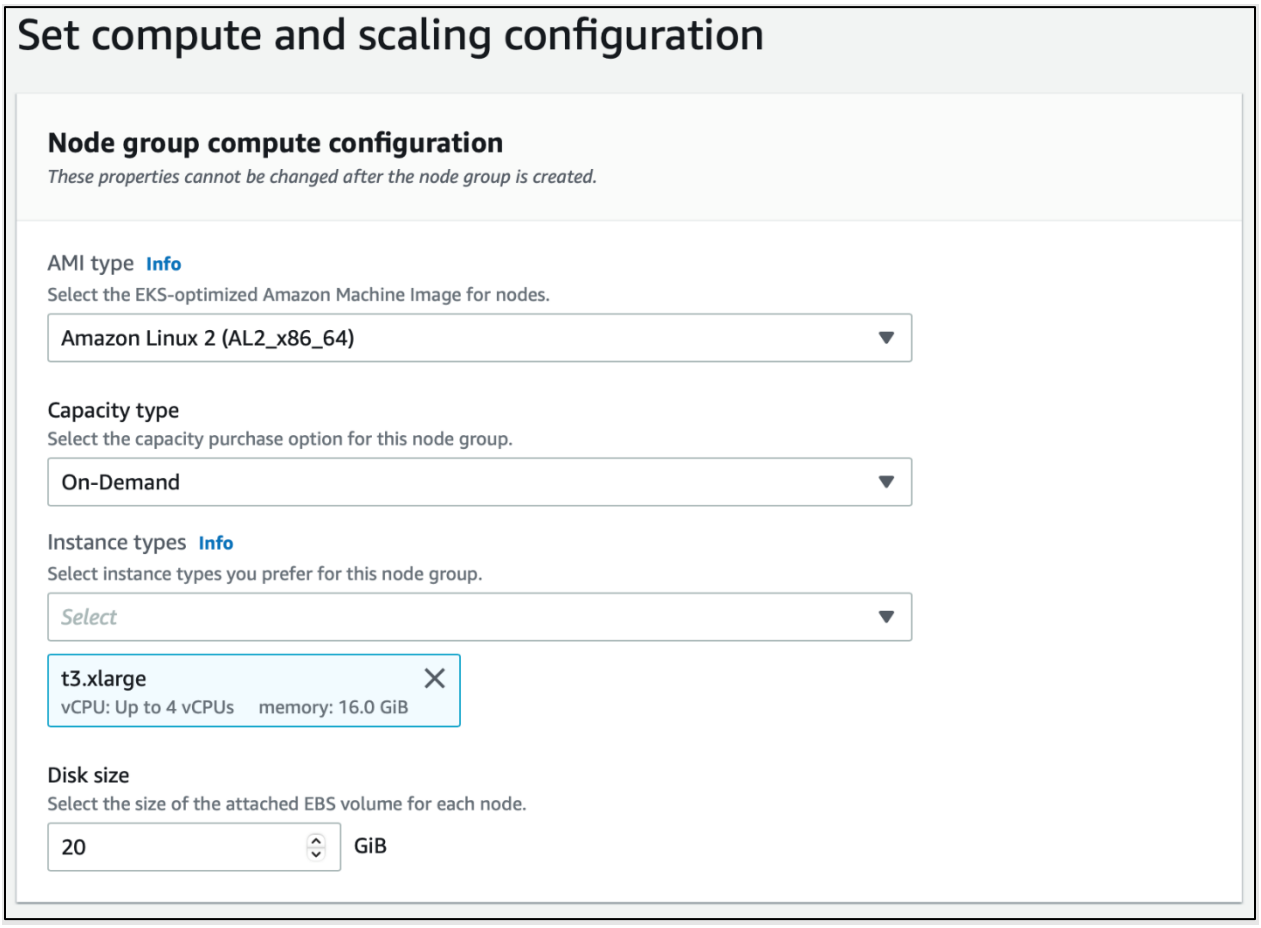




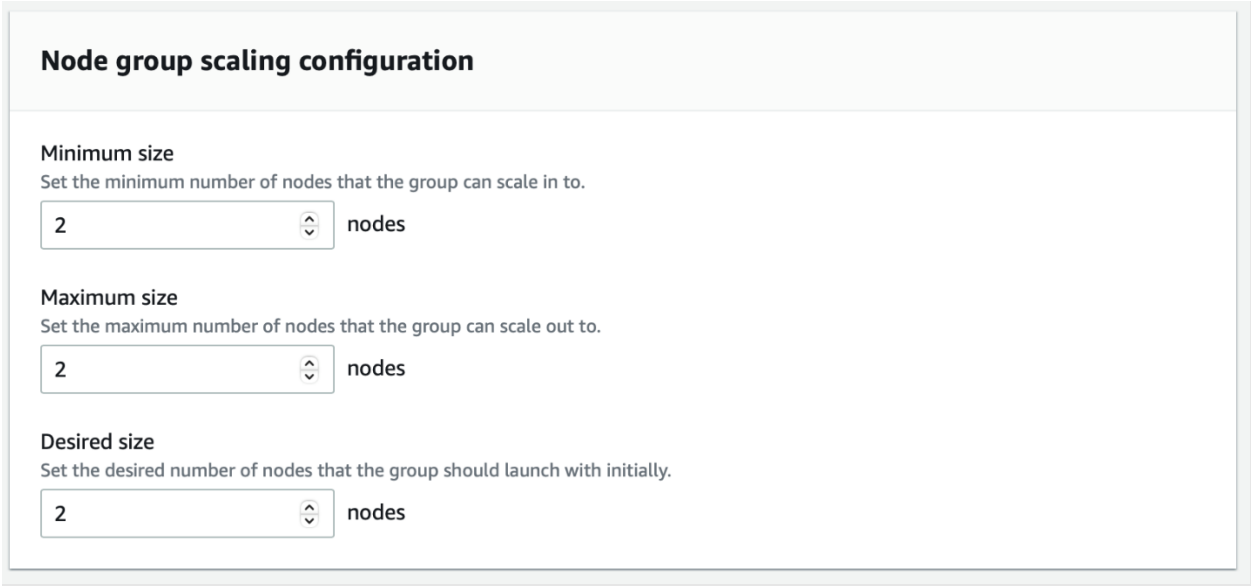
Set compute and scaling configuration

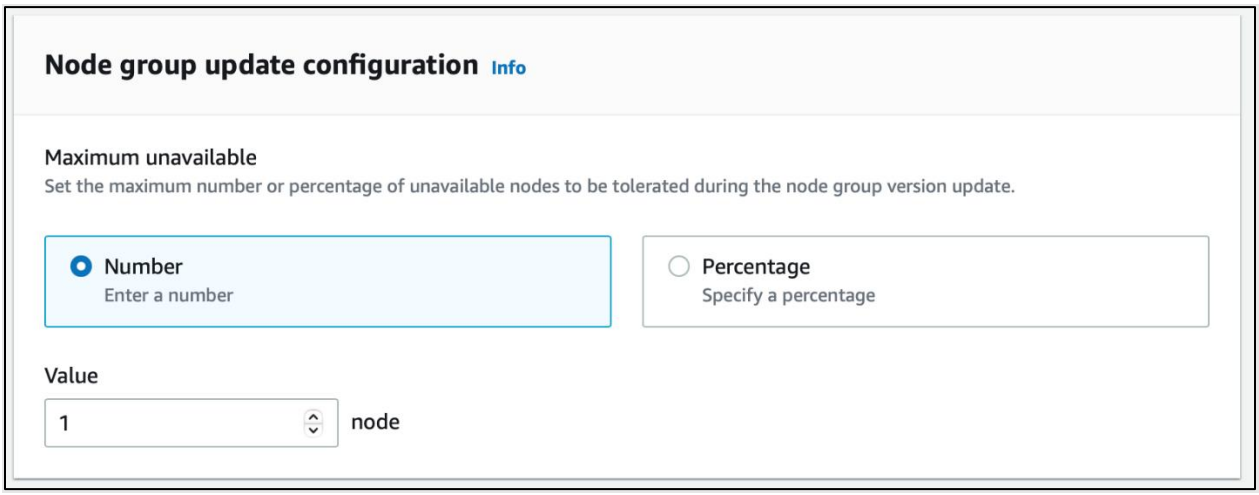
The following default values are recommended however the client should select the operating system that best suits their needs.

NOTE: We recommend instance type be set to: *m6a.2xlarge*

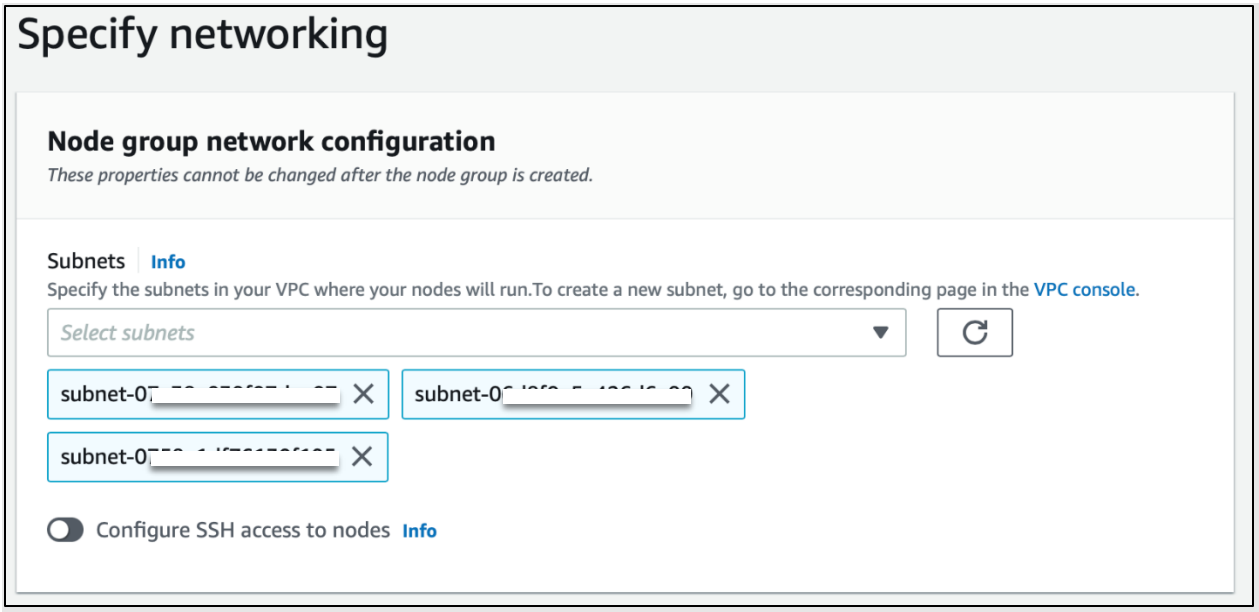


The following can be scaled as be client requirements




Node group update configuration Info
Maximum unavailable
Set the maximum number or percentage of unavailable nodes to be tolerated during the node group version update.
Number
Enter a number
Percentage
Specify a percentage
Value
1
node

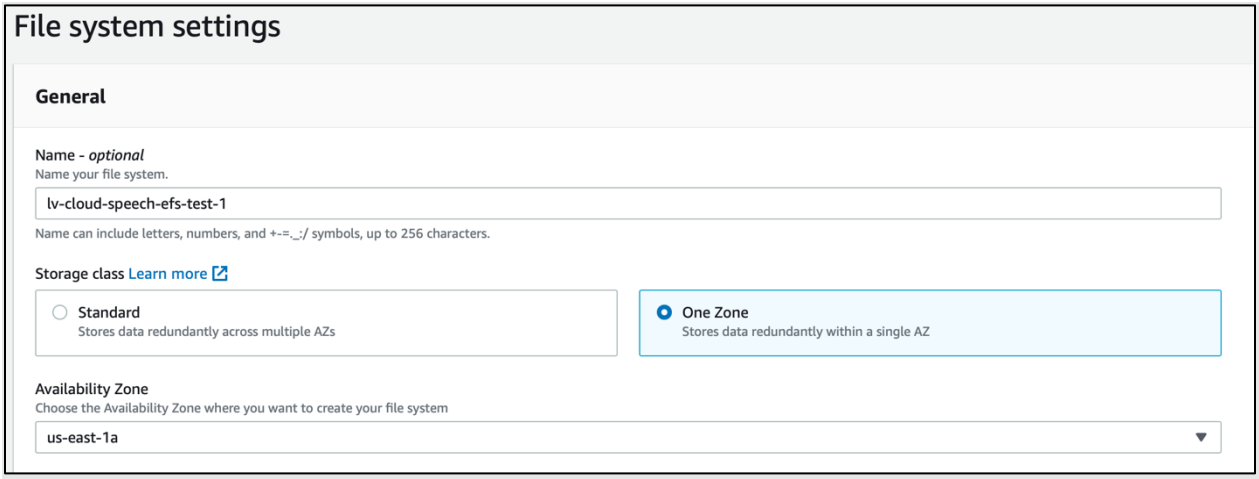
Specify Networking



- Review and create.
- Cluster creation can take several minutes to complete. Monitor for any errors.

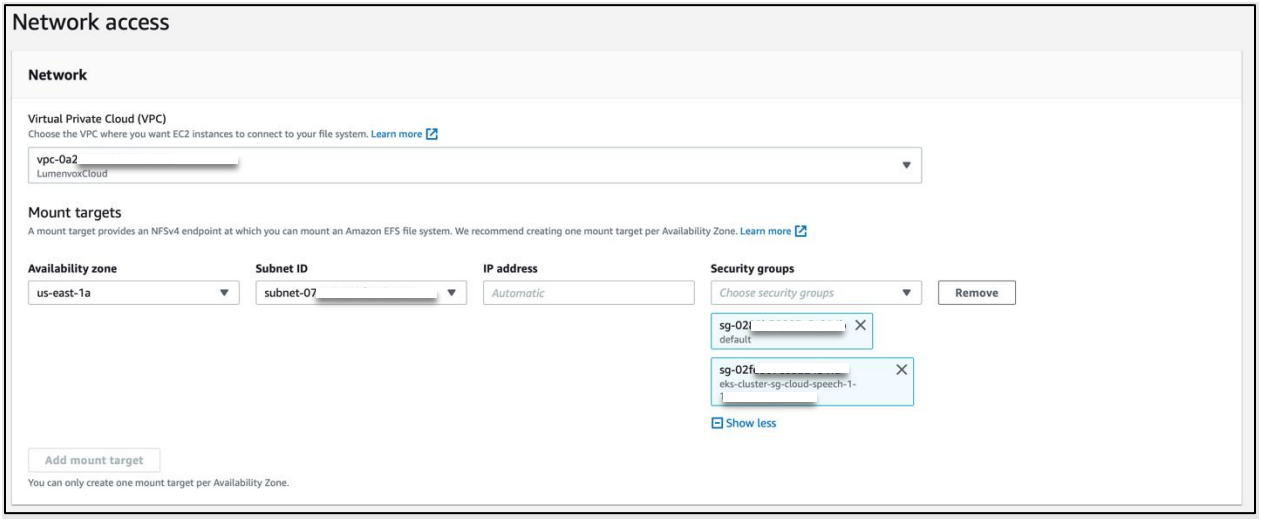
Create EFS Instance (elastic file storage)

Set up as per client requirements, required as the persistent storage device



Network Access

Ensure you select the same VPC as the cluster, also select the default security group as well as the "eks-cluster-sg-***" group created by the cluster



- Review and create
- Create access point
- Click on "Access points" and click on "Create access point" and accept defaults. Then click on attach

Install kubectl

Create access keys

To create access keys for an IAM user

- Sign into the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
- In the navigation pane, choose **Users**.
- Choose the name of the user whose access keys you want to create, and then choose the **Security credentials** tab.
- In the **Access keys** section, choose **Create access key**.
- To view the new access key pair, choose **Show**. You will not have access to the secret access key again after this dialog box closes. Your credentials will look something like this:
    - Access key ID: AKIAIOSFODNN7EXAMPLE
    - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
- To download the key pair, choose **Download .csv file**. Store the keys in a secure location. You will not have access to the secret access key again after this dialog box closes.Keep

the keys confidential in order to protect your AWS account and never email them. Do not share them outside your organization, even if an inquiry appears to come from AWS or Amazon.com. No one who legitimately represents Amazon will ever ask you for your secret key.

- After you download the .csv file, choose **Close**. When you create an access key, the key pair is active by default, and you can use the pair right away.

Install AWS CLI

https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Check Version

```
1. aws --version
```

Check current identity

```
1. aws sts get-caller-identity
```

If you need to modify the identity use the commands below:

```
1. aws configure
2. AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
3. AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
4. Default region name [None]: us-east-1
5. Default output format [None]: json
```

Update kubeconfig

```
1. aws eks --region us-east-1 update-kubeconfig --name cloud-speech-1
```

Install Helm

```
1. curl -fsSL -o get_helm.sh https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
2. chmod 700 get_helm.sh
3. ./get_helm.sh
```

Install Linkerd CLI

```
1. curl --proto '=https' --tlsv1.2 -sSfL https://run.linkerd.io/install | sh
2. Add the Linkerd CLI to your path
3. export PATH=$PATH:$HOME/.linkerd2/bin
```

Install Linkerd

```
1. Linkerd check –pre
2. Linkerd install –crds | kubectl apply –f -
3. linkerd install --set proxyInit.runAsRoot=true | kubectl apply -f -
4. linkerd check
```

Install Jaeger

```
1. linkerd jaeger install | kubectl apply -f -
```

Install Linkerd viz Dashboard

```
1. linkerd viz install | kubectl apply -f –
```

Should be ever need to uninstall Linkerd for troubleshooting purposes (e.g. deleting a cluster or you have issues with the Linkerd side cars you can use the following commands:

Uninstall Linkerd

```
1. linkerd viz uninstall | kubectl delete -f -
2. linkerd jaeger uninstall | kubectl delete -f -
3. linkerd uninstall | kubectl delete -f -
```

Install nginx Ingress

```
1. helm repo add ingress-nginx https://kubernetes.github.io/ingress-nginx
2. helm repo update
3. kubectl create ns ingress-nginx
4. helm upgrade --install ingress-nginx ingress-nginx --repo https://kubernetes.github.io/ingress-nginx -n ingress-nginx --create-namespace --set controller.hostNetwork=true --set controller.allowSnippetAnnotations=true
```

Create LumenVox Namespace

```
1. kubectl create namespace lumenvox
```

Change NameSpace

```
1. kubectl config set-context --current --namespace=lumenvox
```

© 2024 LumenVox

Setup TLS for Ingress

```
1. openssl genrsa -out server.key 2048
2. openssl req -new -x509 -sha256 -key server.key -out server.crt -days 3650 -addext "subjectAltName
= DNS:lumenvox-api.testmachine.com, DNS:biometric-api.testmachine.com, DNS:management-
api.testmachine.com, DNS:reporting-api.testmachine.com, DNS:admin-portal.testmachine.com,
DNS:deployment-portal.testmachine.com"
3. kubectl create secret tls speech-tls-secret --key server.key --cert server.crt
```

Apply Secrets File

```
1. kubectl apply -f lumenvox-secrets.yaml
```

Configure Helm

```
1. helm repo add lumenvox https://lumenvox.github.io/helm-charts
2. helm repo update
```

Deploy Public Chart (Internal Resources)

```
1. helm install lumenvox lumenvox/lumenvox -f aws-speech-values-internal-resources.yaml -n lumenvox
```

See Appendix 4 for final steps to for complete installation by setting up a deployment

© 2024 LumenVox

APPENDIX 3 – LV Containers Quick Start (kubeadm)

Clients wanting to perform an on-premises Kubernetes installation can follow these steps for Linux (Some steps may differ to various Linux installations). This makes use of kubeadm to simply the installation process for clients.

Getting started

Access the following page GitHub - lumenvox/containers-quick-start: Setup scripts for LumenVox Containers to obtain the required installation files, steps, prerequisites, and supported environments.

Once the required operating environment has been set up the following steps can be followed by running the following in the command line:

Install LumenVox Containers Repo

git clone https://github.com/lumenvox/containers-quick-start.git

cd containers-quick-start/

Generate SSL Certificate

```
1. openssl genrsa -out server.key 2048
2. openssl req -new -x509 -sha256 -key server.key -out server.crt -days 3650 -addext "subjectAltName
= DNS:lumenvox-api.testmachine.com, DNS:biometric-api.testmachine.com, DNS:management-
api.testmachine.com, DNS:reporting-api.testmachine.com, DNS:admin-portal.testmachine.com,
DNS:deployment-portal.testmachine.com"
```

The subject alternative name can be specific to the customers environment.

Grant execute permissions to scripts

```
1. chmod +x *.sh
```

Perform Installation

Edit values.yaml file as per installation requirements.
./lumenvox-control-install.sh values.yaml server.key server.crt (use command as is)

Document the passwords created for redistribution, RabbitMQ, mongo and Postgres as these will be used later to create the deployment.

The following is an example of the values.yaml file

```yaml
# kubeadm basic values

lumenvox-common:
  licensing:
    # Note: licensing (and therefore system) will not work without a valid clusterGuid value
    clusterGuid: "LUMENVOX-TO-PROVICE-CLUSTER_GUID"
  rabbitmq:
    enabled: false
  redis:
    enabled: false
  mongodb:
    enabled: false
  postgresql:
    enabled: false

global:
  defaultNamespace: "lumenvox"
  hostnameSuffix: "testmachine.com"
  lumenvox:
    ingress:
      className: nginx
    loggingVerbosity: "debug"
  image:
    pullPolicy: Always
    tag: ":4.6.0"
  mockData:
    enabled: false
    testDataString: ""
    deploymentId: []
    jsonKeys:
      master: ''
      customer: ''
      emergencyPublic: ''
  rabbitmq:
    enableTLS: false
    connection:
      url: "192.168.1.188"
  redis:
    enableTLS: false
    connection:
      url: "192.168.1.188"
  mongodb:
    connection:
      url: "192.168.1.188"
  postgresql:
    connection:
      url: "192.168.1.188"
      databaseName: "lumenvox_single_db"
  enabled:
    lumenvoxSpeech: true
    lumenvoxVb: false
    lumenvoxCommon: true
  enableItn: false
  asrLanguages:
    - name: "en"
  ttsLanguages:
    - name: "en_us"
      voices:
        - name: "chris"
  vbLanguages:
    - name: "en_US"
```

```
62.       version: "2.1.15"
```

See Appendix 4 for final steps to for complete installation by setting up a deployment

Renewal of certificates

Kubeadm creates a certificate upon initial installation and this certificate expires every 365 days and needs to be manually renewed before expiration. The following steps can be followed:

Backing up the old certs and configs

```
1. mkdir -p $HOME/k8s-old-certs/pki
2. sudo /bin/cp -p /etc/kubernetes/pki/*.* $HOME/k8s-old-certs/pki
3. sudo /bin/cp -p /etc/kubernetes/*.conf $HOME/k8s-old-certs
4. mkdir -p $HOME/k8s-old-certs/.kube
5. sudo /bin/cp -p ~/.kube/config $HOME/k8s-old-certs/.kube/.
```

Renewing the certificates

```
1. sudo kubeadm certs renew all
```

This is the output from running the command:

```
[renew] Reading configuration from the cluster...
[renew] FYI: You can look at this config file with 'kubectl -n kube-system get cm kubeadm-config -o yaml'
[renew] Error reading configuration from the Cluster. Falling back to default configuration

certificate embedded in the kubeconfig file for the admin to use and for kubeadm itself renewed
certificate for serving the Kubernetes API renewed
certificate the apiserver uses to access etcd renewed
certificate for the API server to connect to kubelet renewed
certificate embedded in the kubeconfig file for the controller manager to use renewed
certificate for liveness probes to healthcheck etcd renewed
certificate for etcd nodes to communicate with each other renewed
certificate for serving etcd renewed
certificate for the front proxy client renewed
certificate embedded in the kubeconfig file for the scheduler manager to use renewed
```

The certificate used by kubelet

You'll find four files /var/lib/kubelet/pki/. One of them is kubelet.crt. This file has also expired if you check with openssl:

```
1. sudo cat /var/lib/kubelet/pki/kubelet.crt | openssl x509 -noout -enddate
```

Deleting old certificates

Stopping kubectl was not mentioned in any of the articles we suggest it gets done:

```
1. sudo systemctl stop kubelet
2. sudo rm /etc/kubernetes/kubelet.conf
3. sudo ls /var/lib/kubelet/pki
4.
5. sudo rm /var/lib/kubelet/pki/kubelet-client-<filename-from-ls-command>.pem
6. sudo rm /var/lib/kubelet/pki/kubelet-client-current.pem
7. sudo rm /var/lib/kubelet/pki/kubelet.crt
8. sudo rm /var/lib/kubelet/pki/kubelet.key
```

Fixing Kubelet Service

This specific command regenerates the kube config file. The only article which mentions this step is [this one](#).

```
1. sudo kubeadm init phase kubeconfig kubelet
2. sudo systemctl start kubelet
```

Updating the client data configs

```
1. sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
2. sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

APPENDIX 4: Setting up a deployment

The following steps must be done across all forms of deployments:

Ensure that the following host entries have been added to the machine that you will be accessing the Portal / API's from:

The following is an example of a setup for a test environment:

Kubeadm (IP address is that of the VM running the LumenVox container deployment)
192.168.1.103 biometric-api.testmachine.com
192.168.1.103 reporting-api.testmachine.com
192.168.1.103 lumenvox-api.testmachine.com
192.168.1.103 admin-portal.testmachine.com
192.168.1.103 management-api.testmachine.com
192.168.1.103 deployment-portal.testmachine.com
192.168.1.103 reporting-bio-api.testmachine.com

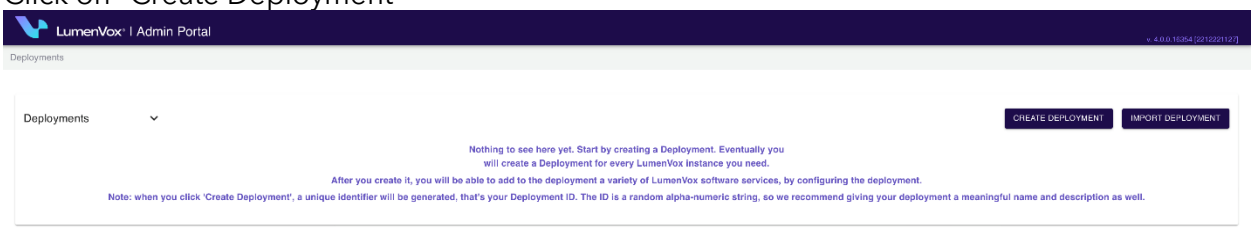
Open a browser session to:

https://admin-portal.testmachine.com

Configure set up of deployment information

Create deployment

Click on "Create Deployment"

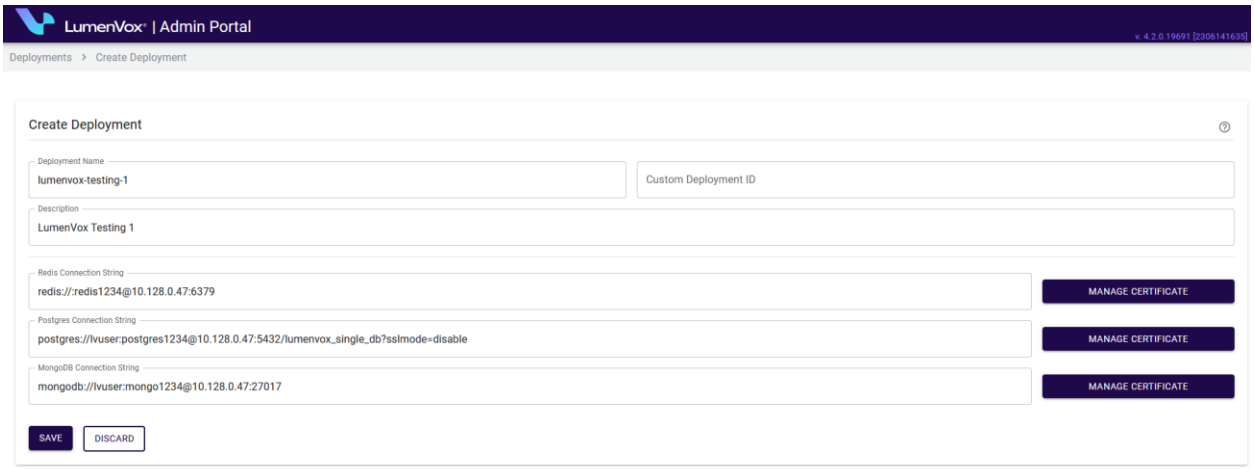


The following are examples of connection strings for the external services:

Redis:
redis://:redis1234@10.128.0.47:6379

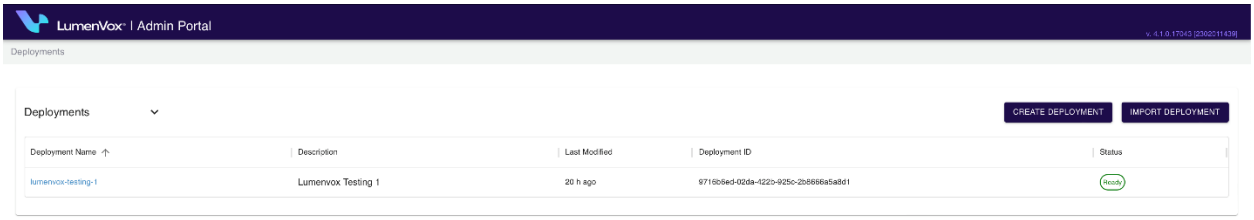
PostgreSQL:
postgres://lvuser:postgres1234@10.128.0.47:5432/lumenvox_single_db?sslmode=disable

MongoDB:
mongodb://lvuser:mongo1234@10.128.0.47:27017



Complete the fields as required. “Custom Deployment ID” is not a mandatory field (if left blank a random GUID will be generated)
Insert required connection strings

Click “Save”



APPENDIX 5 – Basic KUBECTL Commands

The following KubeCTL commands are the most common ones that a client may use

Get list of pods

```
1. Kubectl get pods
```

Get status of each pod

```
1. Kubectl describe pod {pod name}
```

Get logs for a pod

To look into the actual pod e.g. deployment which is the most important (start here for any troubleshooting on installation)

```
1. Kubectl logs {insert pod name} -c [pod component}
```

Show ingress

Used to see what ingress points are configured, hostname configures and ip addresses assigned

```
1. Kubectl get ingress
```

To remove a pod

This is used to delete a pod so that it can be recreated

```
1. kubectl delete pod {Insert pod name}
```

To scale a pod

This command can be used to instruct Kubernetes to scale a specific pod if auto scaling is not enabled.

```
1. kubectl scale deploy {insert pod name} --replicas=2
```

© 2024 LumenVox

APPENDIX 6 - Sample secrets file

This file should be applied to your Kubernetes cluster before installing any LumenVox Helm Chart. This assigns credentials for various services. You can change these to meet your specific installation requirements. Adjust namespace values if needed.

```
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: rabbitmq-existing-secret
type: Opaque
data:
  rabbitmq-password: EXAMPLE241901SNSJKvzw2
---
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: postgres-existing-secret
type: Opaque
data:
  postgresql-password: EXAMPLE241901SNSJKvzw2
  postgresql-postgres-password: EXAMPLE241901SNSJKvzw2
---
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: redis-existing-secret
type: Opaque
data:
  redis-password: EXAMPLE241901SNSJKvzw2
---
apiVersion: v1
kind: Secret
metadata:
  namespace: lumenvox
  name: mongodb-existing-secret
type: Opaque
data:
  mongodb-root-password: EXAMPLE241901SNSJKvzw2
---

```

APPENDIX 7 - MRCP API Server

Installation

The minimum server requirements are as follows:

- Linux OS capable of running Docker and Docker Compose plugin
- CPU – 2 CPU Cores
- Memory – 4GB Memory
- Boot Disk – 20GB

The following steps must be followed for a MRCP installation:

These notes were for an installation performed on Ubuntu Server.

```
1. sudo apt-get update
2. sudo apt-get upgrade
```

Install Docker

```
1. sudo apt-get install ca-certificates curl gnupg lsb-release
2. sudo mkdir -p /etc/apt/keyrings
3. curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
4. echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg]
https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" | sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null
5. sudo apt-get update
6. sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

Setup docker to start automatically after server reboot

```
1. sudo systemctl enable docker
2. sudo systemctl start docker
3. sudo groupadd docker
4. sudo usermod -aG docker $USER
```

Logout and login

If you receive this error then ====Permission denied when trying to connect to Docker Daemon

```
1. sudo chmod 666 /var/run/docker.sock
```

Install mrcp-api
git clone https://github.com/lumenvox/mrcp-api.git

```
1. cd mrcp-api/docker/
```

If pointing to a lumenvox-api instance running with TLS follow these steps

```
1. cd mrcp-api/docker/
2. mkdir certs
```

copy server.crt file from instance running lumenvox-api to mrcp-api/docker/certs/

Edit .env file

Make the necessary edits to the .env file as per your setup

Example

```
 1. APPLICATION_DOMAIN=testmachine.com
 2.
 3. MEDIA_SERVER__ALLOW_DEPLOYMENT_OVERRIDE=false
 4. MEDIA_SERVER__ALLOW_OPERATOR_OVERRIDE=false
 5. MEDIA_SERVER__DEFAULT_DEPLOYMENT_ID=your deployment id
 6. MEDIA_SERVER__DEFAULT_OPERATOR_ID=00000000-0000-0000-0000-000000000000
 7. MEDIA_SERVER__SERVER_IP=the IP address of the server running the MRCP API instance
 8. MEDIA_SERVER__SPEECH_API_ADDRESS=lumenvox-api.testmachine.com
 9. MEDIA_SERVER__SPEECH_API_PORT=443
10. MEDIA_SERVER__USE_TLS=true
```

Value to configure hostname mapping. If you don't have a registered domain for your speech API, you should set the mapping here. If you do have a registered domain, you can set this to empty or comment it out entirely.
The hostname should match the value of MEDIA_SERVER__SPEECH_API_ADDRESS.
MEDIA_SERVER__HOST_MAP=lumenvox-api.testmachine.com:ip address of the speech-api.testmachine.com interface

Edit hosts file

Edit the /ect/hosts file with an entry like the example below:
ip address of the lumenvox-api.testmachine.com interface lumenvox-api.testmachine.com

Launch Docker Image

```
1. docker compose up -d
```

NB: MRCP services will be deployment-specific and there is no option to change the deployment ID used by the service. If customers want to provide different tenants (deployments) access to MRCP, each will require their own MRCP service configured for their own deploymentId.

Server Ports Setup

The MRCP API Server is responsible for providing connectivity between various platforms that use MRCP to connect to the LumenVox speech services. Typically when connecting to the LumenVox Server, these platforms would use either SIP or RTSP sessions to negotiate the parameters of the connection, including which MRCP port and RTP ports would be used. Either

of the SIP and RTSP ports can be disabled by setting the port value to 0 if not required, although leaving the port enabled does not pose much of an overhead.

LumenVox supports SIP connections using either UDP or TCP protocols, so be sure to configure the appropriate setting for this port when setting up your firewall rules. Also note that often when LumenVox is installed on the same server as another platform that uses SIP connectivity, there may be a port conflict between the platform and the LumenVox MRCP Server, since both are trying to use port 5060 by default, so many times it is easier to change the MRCP Server SIP port from the default value to something else (5066 for example).

For RTP connections: we allow configuration of the port ranges for MRCP and RTP connectivity to avoid overlapping any port range used by other applications. RTP data is typically inbound to the MRCP Server for ASR audio, and outbound from the MRCP Server for TTS audio.

Name	Default Port / Range	Protocol	Direction	Configuration Setting
MRCP Connectivity	20000 - 24999	TCP	IN	media_server.conf / [GLOBAL] mrcp_server_port_base
RTP audio	25000 - 29999	UDP	IN/OUT	media_server.conf / [GLOBAL] rtp_server_port_base
SIP Port	5060	UDP/TCP	IN	media_server.conf / [GLOBAL] sip_port
RTSP Port	554	TCP	IN	media_server.conf / [GLOBAL] rtsp_port

APPENDIX 8 – Security set up considerations

Firewalls

The following firewalls & ports need to be opened by security team:

https://licensing.lumenvox.com via port 443.

https://lumenvox-public-assets.s3.amazonaws.com/model-files/*

Firewall rules are also required for the admin & deployment portals e.g.

- admin-portal.testmachine.com
- deployment-portal.testmachine.com

Cluster GUID

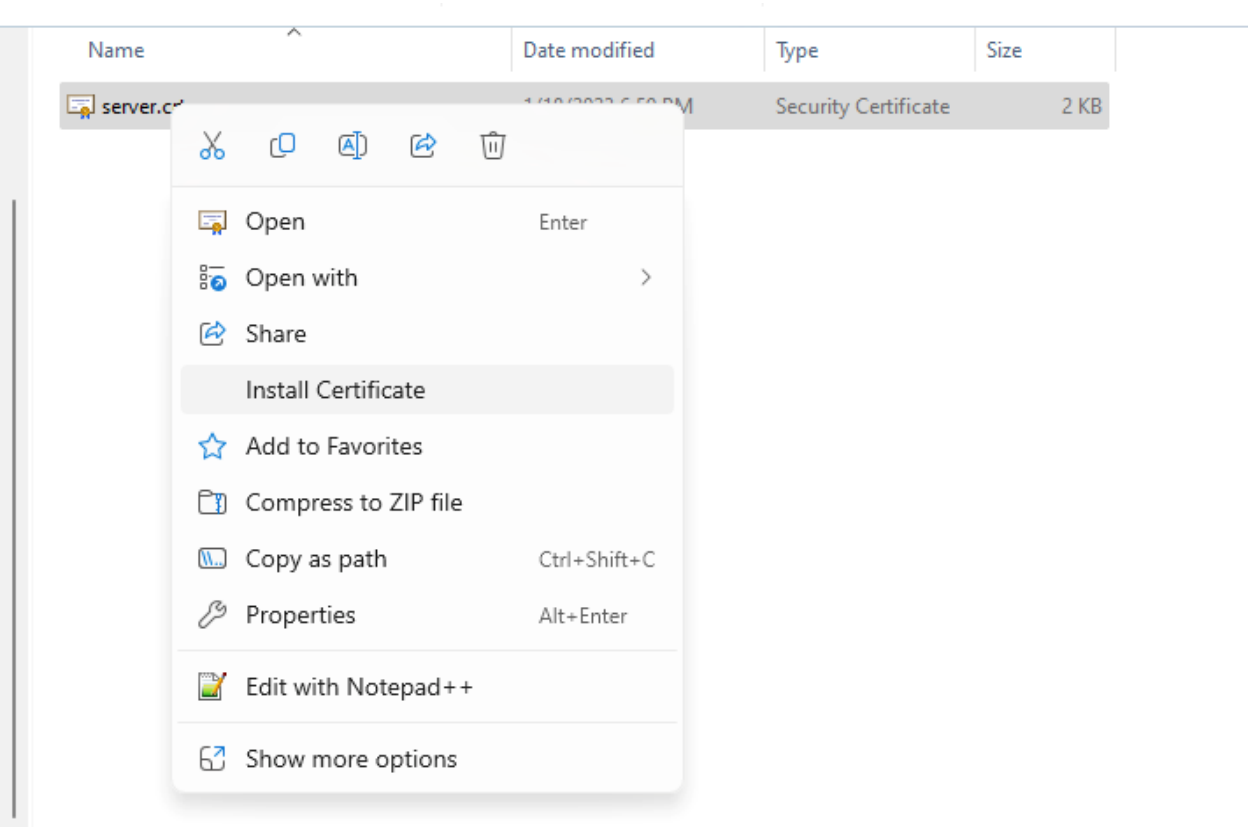
The cluster encryption master key gets created when the deployment service starts for the first time and is stored in the database. The cluster encryption master key is encrypted with the cluster GUID. The customer keys are then created for each deployment. These are encrypted using the cluster encryption master key.

APPENDIX 9 – setting up self-signed certificates for

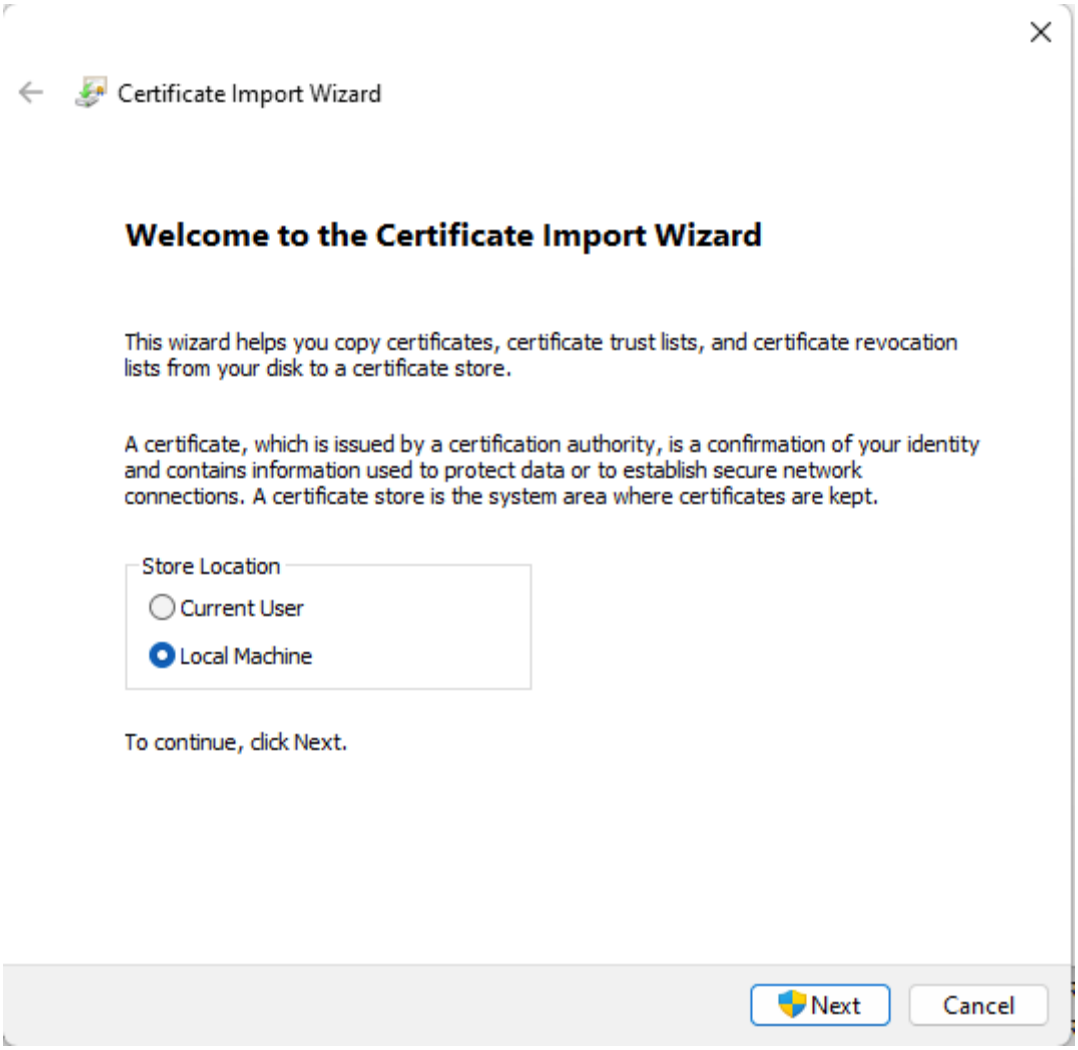
The following steps should be taken if you have used a self-signed certificate in your LumenVox software installation and you are wanting to access the admin or deployment portal from your desktop machine.

The certificate is the one generated when installing the LumenVox Software. Copy this certificate into a folder on your local machine.

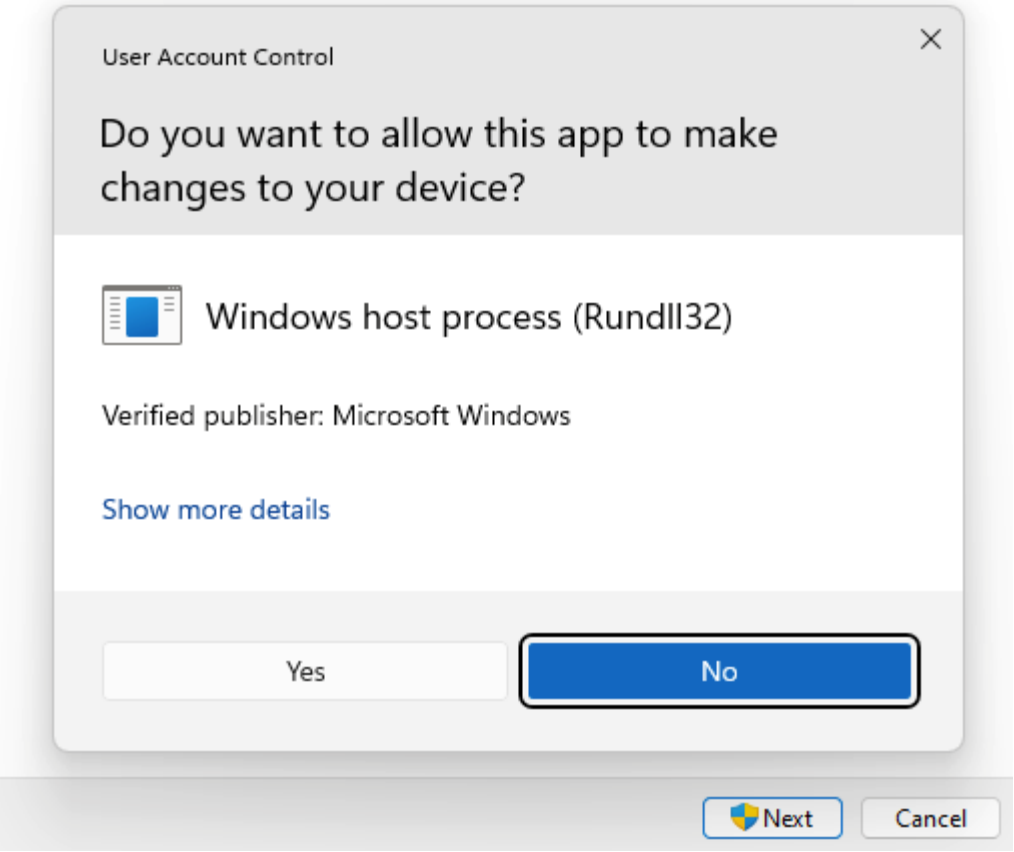
Right Click on your certificate file and select “Install Certificate”



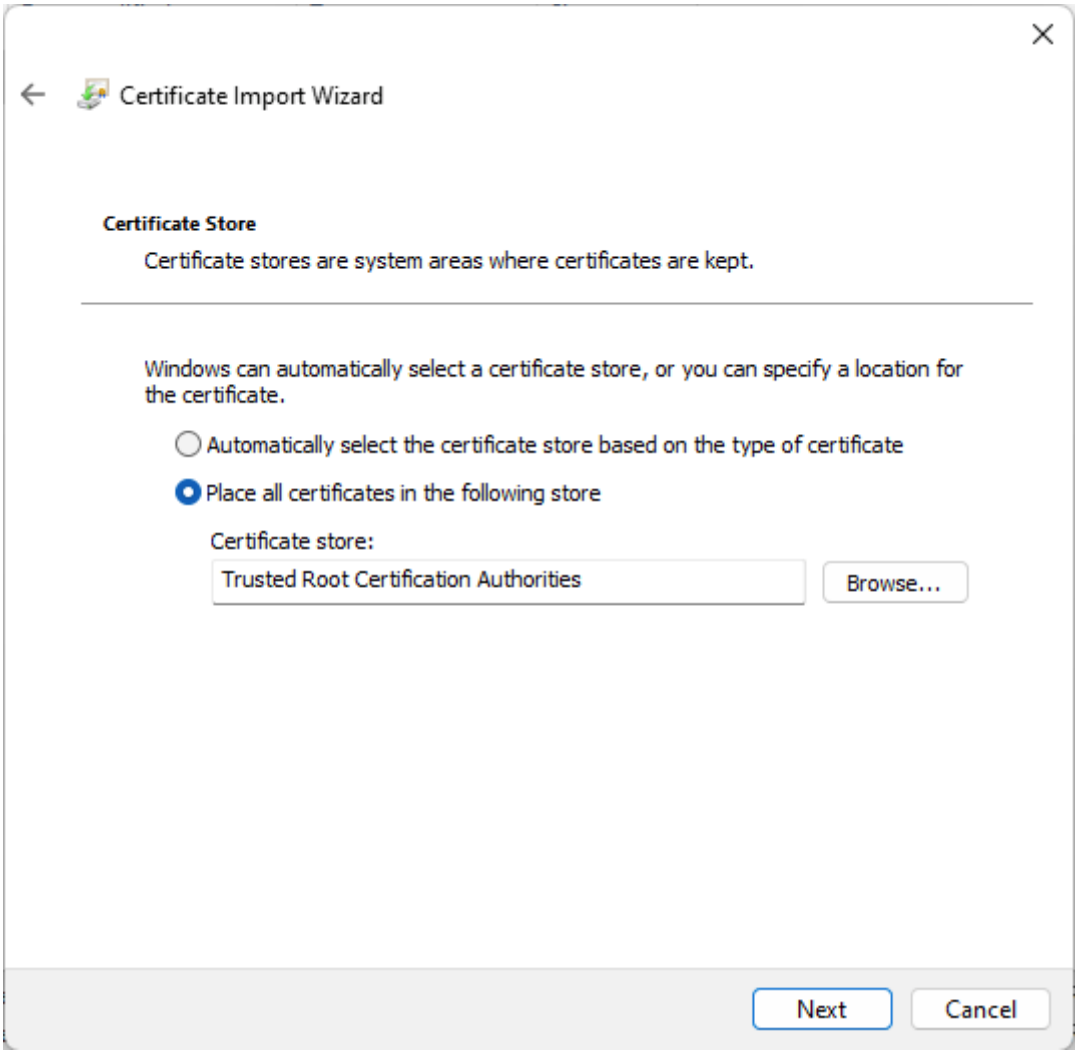
Select “Local Machine” and click “Next”



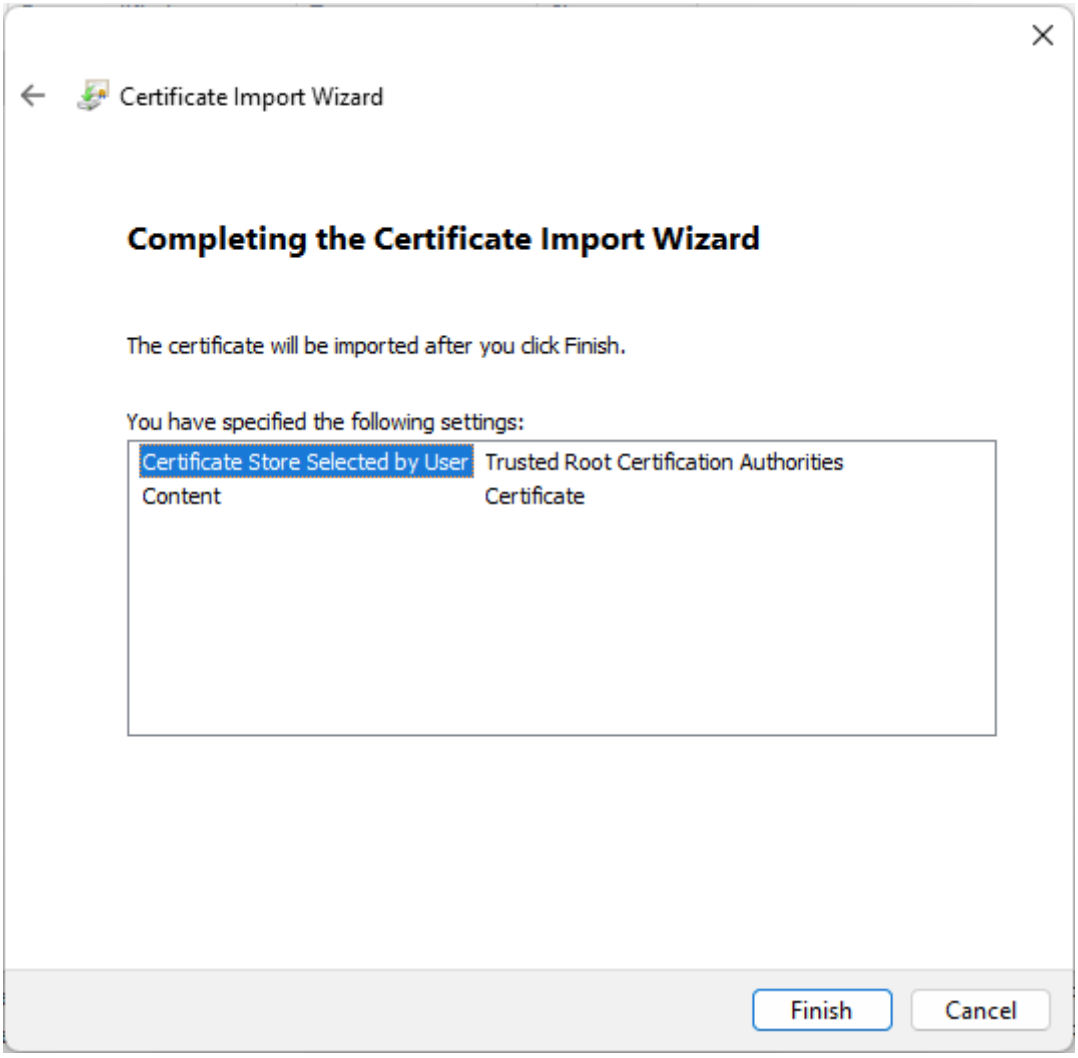
If presented with a User Account Control windows click “Yes”



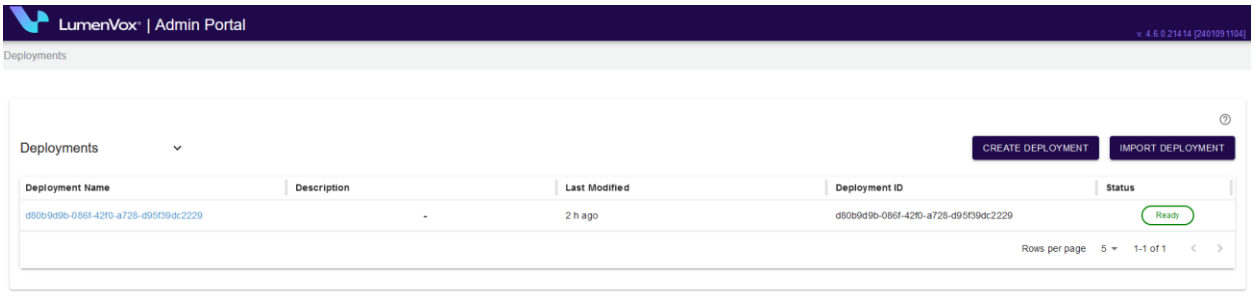
Select the option “Place all certificates in the following store” and select the “Trusted Root Certification Authorities” store and click Next



Click Finish



Now when you open a browser (you may have to open a private/incognito tab because of cache) because the certificate is trusted you will be able to access https://admin-portal.testmachine (or the equivalent domain name you created when you generated the certificate) without any security prompts or blank screens.



LumenVox®

About LumenVox

LumenVox transforms customer communication. Our flexible and cost-effective technology enables you to create effortless, secure self-service and customer-agent interactions. We provide a complete suite of speech and authentication technology to make customer relations faster, stronger and safer than ever before. Our expertise is extensive— we support a multitude of applications for speech and voice biometrics. And we do it all by putting you and your customers first.

Interested in finding out more about this product?

Learn More →

Contact

LVsales@lumenvox.com +1 (858) 707-7700

www.LumenVox.com