Cloud 4.0

This contains documentation for Cloud 4.0, a cloud based on microservice architecture

Development Notes & Links

Overview

This page contains a list of links and information pertaining to the development of microservices.

Links

- System Design Primer - Github How to design large scale systems. Includes information on databases, caching and other topics involving creating large, scalable systems
- Design Microservices the Right Way - Youtube - Michel, former CEO of flow.io walks us through how his company used OpenAPI and other tools to develop their microservices. He also talks about using Apache Kafka for messaging in his microservice world. A very large inspiration for the entire project
- Authentication as a Microservice - A good talk on how to have a microservice that authenticates and authorizes users
- Roles vs Scopes from Stackoverflow - Good explanation on Roles and Scopes
- Spring Boot Apps with Kong - Good resource on auto generating a kong configuration
- Microsoft API Development Guidelines - Great resource of how Microsoft designs their Rest APIs
-

README

What is Cloud 4.0

Cloud 4.0 aims to centralize and integrate many of the applications found in the docker-compose stack. This will allow for users to have a unified experience across many different services and improve synergy amongst a variety of different types of services.

Although the downtime for these services should be **minimal.** Developers who are relying on these services should plan for these services to go down at any time.

Services Being Replaced in Cloud 4.0

- Bitwarden
- Nextcloud
- Password Self Service
- Wordpress (Frontend and Blog)
- Guacamole

Services Being Added in Cloud 4.0

- Messaging/Chat
- Registry Service
- General developer API
- KeyCloak

Services Not Removed in Cloud 4.0

- Gitea
- Fusion ldap Directory
- Bookstack

Microservice Planning and Specifications

This chapter contains all specification documents for home made microservices

Base Microservice Specifications

This document will describe basic functions all Microservices will have. The service's endpoints and controller files will be generated using OpenAPI's generator. As such, all microservices will follow OpenAPI's specification for code generation and documentation.

Software Topics

Base Image

The official Debian docker image will be used for all containers since the dev container will also be a Debian based image. The latest Java version will also be used for running the microservices

Versioning

To distinguish when containers were built, each container will have a buildinfo.json text file generated.

```
{
  "timestamp":"2021-10-16T14:45:51.208015",
  "base-version":"<insert base version here>",
  "application-version":"<insert application version here>"
}
```

In the keys "base-version" and "application-version", developers will be required to follow the semantic versioning method, a sequence versioning method that follows a vX.Y.Z format (eg: v1.2.3), where X is a major change, Y is a minor change and Z is a patch change. Additional information can be added on the end of the version number to distinguish between test builds, nightly builds, ect. Example: v1.1.2-20211209Nightly.

Logging

In effort to centralize logging, all containers must log to the standard out of the docker-compose process. (This shouldn't be an issue since if you just log everything to STD out, it'll log to the docker logs). One consideration to make is that logs will be sent to a centralized logging server so they should be easily grep-able.

Event Messaging

Microservices will have their own event messaging topics that other microservices can subscribe to. For instance, an authentication microservice might have a topic for notifying other services when tokens have expired.

Database/caching

Each microservice will have its own cache. There will be a group of other services that provide database functionality.

Well Known Endpoints

/health

Can return HTTP status 200 for healthy or 500 for not healthy.

This will not be available to end users

/docs

HTTP documentation generated with OpenAPI. **This will not be available to end users**

/build

Information about the application.

/operations

Returns information about running operations.

Consider the following, client POSTs a create endpoint to create an ambiguous object that take a long time to construct. The client will receive the following response if the operation is created:

```
202 Accepted


{
"operationLocation": "https://example.com/operations/<operation-id>"
}
```

The client can then invoke a GET request on the operationLocation to get the status of the operation. If the server responds that the operation isn't ready, a Retry-After HTTP header will be sent.

```
200 OK
Retry-After: 30


{
```

```
  "createdDateTime": "2022-09-20T15:00:00.00Z",
  "status": "running",
  "estimatedCompleteTime: "2022-09-20T15:01:00.00Z"
}
```

```
200 OK
Retry-After: 30

{
  "createdDateTime": "2015-06-19T12-01-03.4Z",
  "status": "running",
  "percentComplete": 0,
}
```

When the request is complete, the operation will respond with the resource's location and other data

```
HTTP 200

{
  "createdDateTime": "2022-09-20T15:00:00.00Z",
  "lastActionDateTime": "2022-09-20T15:01:00.00Z",
  "status": "succeeded",
  "resourceLocation": "https://example.com/<some-resource>/<some-resource-id>"
}
```

API Guidelines

All microservices will be required to follow the guidelines as closely as possible. This is to ensure stability of the ecosystem and to make developing client applications easier. The API guidelines will be a combination of the following existing guidelines:

- Microsoft's API Guidelines

Errors and Faults

An error occurs when the client passes invalid data to the server. Typically, errors are 4XX errors. When an error occurs, the service should be able to operate normally and not shut down.

A fault occurs when the service can not pass the correct response to the clients. Typically, faults are 5XX HTTP errors. Services should attempt to recover from a fault in order to avoid shutting down, however if they can not recover, the health of the service should be labeled as Unhealthy so

it can be troubleshooted/restarted.

URL Structures

URLs should be easy to read by people. This includes the maximum length of a URL. While RFC7230 does not a specific URL length, the length of a URL should not exceed 2,083 characters (This is the maximum length for Internet Explorer).

Required Supported Methods

Services should support the required methods with no exception: GET, POST, DELETE, PUT, HEAD, OPTIONS. If Open API is used, all of these methods will be available for use. Below summarizes a table of the behavior of the methods on each resource from an ecommerce point of view:

Resource	POST	GET	PUT	DELETE
/customers	Create a new customer	Retrieve all customers	Bulk update of customers	Remove all customers
/customers/1	Error	Retrieve the details for customer 1	Update the details of customer 1 if it exists	Remove customer 1
/customers/1/orders	Create a new order for customer 1	Retrieve all orders for customer 1	Bulk update of orders for customer 1	Remove all orders for customer 1

POST Method

POST should be used for creating resources. A POST request should always return HTTP 201 (Created) with the new location of the object.

For example, consider an API for managing users, the client can POST data a URL to create users:

```
curl \
  -X POST \
  -d "{'user':'bob12345'}" \
  -H "Content-Type: application/json"
```

Should return

```
201 Created


{
"location":"http://example.com/users/bob12345"
}
```

Additionally, the server may return the entire metadata for the object. The metadata object should have a "self" key to dictate the path to the object.

```
201 Created


{
"username":"bob",
    "self":"http://example.com/users/bob",
    "creation_date":"2022-09-20T15:00:00.00Z"
}
```

Required Request Headers

The following request headers should be sent with every request:

Authorization: This allows the server to authenticate the request. (Note some services can leave this out if they have public facing data)

Date: The date the request was made. The date should follow ISO8601 and should always include a timezone. You can just use Z if using UTC timezone. The server should never assume the clients clock is accurate. If the time is vastly different than the servers time (+ or - 1 minute) the server should throw 403 with a reason of the clock being out of date.

Accept: Allows the server to see what type of content the client accepts

Accept-Encoding: Allows the server to see what encoding the client accepts

Accept-Charset: Allows the server to see what charset the client accepts (This will probably always be UTF-8)

Content-Type: Only used for POST or PUT calls, allows the server to handle incoming data

Prefer: Allows a client to change how much data the server returns. Can either be "return=minimal" or "return=representation"

Required Response Headers

Date: The date the request was sent to the user.

Content-Type: Content type of the response

Content-Encoding: Encoding the server encoded the response to

Optional Response Headers

Retry-After: Tells the client to retry the request after a date in time. This will primarily be used with long running operations. Note that while the HTTP specification states that this can return a number of seconds, for the purpose of the API, only the date will be used. The date should be formatted to fit ISO 8601 standard (eg: YYYY-MM-DDTHH:MM:SSZ format)

Operation-Location: The location of the new operation. The location will be a link

Custom Headers

Custom headers can not be required for a service to work. All services should work within the required headers above. Custom headers should in a generic format or a scoped format. Custom header data should never be JSON objects

Using Query Parameters

Personally identifiable information should never be used in a URLs query parameters. Additionally query parameters should avoid using json objects.

Handling Errors

When an error occurs, the server should return some sort of message. At minimal the error object should contain a "code" key and a "message" key.

```
{
"code":"someCode",
    "message":"some message"
}
```

More complex errors can contain the "target", "details" and "innererror" keys like so:

```
{
"code":"someCode",
    "message":"some message",
    "target": "some target",
    "details": ["some", "details"],
    "innererror": {
   "some complex json object here"
    }
}
```

An example of a more complex error object is:

```
{
  "error": {
```

```
    "code": "BadArgument",
    "message": "Previous passwords may not be reused",
    "target": "password",
    "innererror": {
      "code": "PasswordError",
      "innererror": {
        "code": "PasswordDoesNotMeetPolicy",
        "minLength": "6",
        "maxLength": "64",
        "characterTypes": ["lowerCase","upperCase","number","symbol"],
        "minDistinctCharacterTypes": "2",
        "innererror": {
          "code": "PasswordReuseNotAllowed"
        }
      }
    }
  }
}
```

Special Note on Clients

When services are designed, they should be designed to be effortlessly used by simple HTTP tools such as curl or postman.

Collections

An inevitable topic that will come up during designing will be collections. A collection is just a simple group of similar objects. For example, a group of 'user' objects are 'users', a group of 'file' objects are 'files'. This should be used when designing endpoints.

For example, if you want to get a list of users, you might want to do the following:

```
curl \
  -X GET \
  -H "Content-Type: application/json" \
  https://example.com/users
```

If you want to get a singular user, you would do:

```
curl \
  -X GET \
  -H "Content-Type: application/json" \
```

```
  https://example.com/users/<user-id>
```

For querying and filtering the collection, URL headers should be used. Take care to not expose personal identifiable information!

```
curl \
  -X GET \
  -H "Content-Type: application/json" \
  https://example.com/users?$orderBy=name asc

curl \
  -X GET \
  -H "Content-Type: application/json" \
  https://example.com/users?$filter=name eq 'kyle'&orderBy=createdDate dec
```

Collections will always return a response with a "value" key like so:

```
{
  "value": [
    . . .
  ]
}
```

If no data is found, the HTTP 204 status code should be used

Nested Collections

Nested collections should also be designed around. For example, a user might have multiple friends:

```
curl \
  -X GET \
  -H "Content-Type: application/json" \
  https://example.com/users/<user-id>/friends
```

Nested collections will have to abide by the "value" key rule only if they are retrieved via their nested URL. Eg:

```
curl \
-X GET \
-H "Content-Type: application/json" \
https://example.com/users/BillyTheKid05
```

Will return

```
{
"value": {
    "username":"BillyTheKid05",
        "firstName": "Billy",
        "lastName": "Smith",
        "friends": [
        {
            "username": "JohnApple",
            "firstName": "John",
                "lastName": "Appleseed"
            },
        {
            "username": "PaulB123",
            "firstName": "Paul",
                "lastName": "Bunyon"
            },
            ...
        }
    }
}
```

while

```
curl \
-X GET \
-H "Content-Type: application/json" \
https://example.com/users/BillyTheKid05/friends
```

Will return

```
{
"value": [
        {
            "username": "JohnApple",
            "firstName": "John",
            "lastName": "Appleseed"
         },
        {
```

```
            "username": "PaulB123",
            "firstName": "Paul",
            "lastName": "Bunyon"
         },
         ....
]
}
```

Big Collections and Pagination

Pagination should be used on all results by default. Pagination parameters should be URL query parameters. paginated responses should have a "@nextlink" key that indicates the next link for the user or client to follow. The URL query parameters "skip" and "limit" should be used:

```
{
  "value": [
    . . .
  ]
  "@nextlink":"https://example.com/collection?$orderBy=name asc&skip=10&limit=10"
}
```

If skip either or limit are missing, an HTTP 401 should be sent

Filter Operations

At minimum, the filter should support the following operations:

Operator	Description	Example
Comparison Operators		
eq	Equal	city eq 'Redmond'
ne	Not equal	city ne 'London'
gt	Greater than	price gt 20
ge	Greater than or equal	price ge 10
lt	Less than	price lt 20
le	Less than or equal	price le 100
Logical Operators		
and	Logical and	price le 200 and price gt 3.5
or	Logical or	price le 3.5 or price gt 200
not	Logical negation	not price le 3.5

Operator	Description	Example
Grouping Operators		
( )	Precedence grouping	(priority eq 1 or city eq 'Redmond') and price gt 100

Dates, Times and Intervals

As alluded to above, all dates will follow the ISO 8601 standard. If the date is in UTC time, there should be nothing after the Z

A standard date will look like `YYYY-MM-DD`

A standard date with time will look like `YYYY-MM-DDTHH:MM.SSSZ`

A duration will follow the format: `P[n]Y[n]M[n]DT[n]H[n]M[n]S` where

- P is the duration designator (historically called "period") placed at the start of the duration representation.
- Y is the year designator that follows the value for the number of years.
- M is the month designator that follows the value for the number of months.
- W is the week designator that follows the value for the number of weeks.
- D is the day designator that follows the value for the number of days.
- T is the time designator that precedes the time components of the representation.
- H is the hour designator that follows the value for the number of hours.
- M is the minute designator that follows the value for the number of minutes.
- S is the second designator that follows the value for the number of seconds

Long Running Operations

It is important that users are able to monitor operations that run for a long period of time. When a long run operation is invoked, the client will be able to pol the operation to get an idea of where the task is at. All operations will be given a task id and will be able to be polled at the /operations end point.

Returning Specific Fields

Some clients might not need the entire object that it given. To save on bandwidth, services can implement a fields URL parameter to specify the fields they need. For example, consider a user model:

```
{
  "id":"0",
  "firstName": "Bob",
  "lastName":"Smith",
  "birthday":"20220101",
```

```
  "phone":"(999) 999-9999"
}
```

Let's say you want just the first and last names of the user. You can specify that in the fields URL parameter:

```
curl \
  -X GET \
  -H "Content-Type: application/json" \
  https://example.com/users/0?fields=firstName,lastName
```

This will return

```
{
  "firstName": "Bob",
  "lastName":"Smith"
}
```

Likewise, on a collection:

```
curl \
  -X GET \
  -H "Content-Type: application/json" \
  https://example.com/users?fields=firstName,lastName
```

```
{
"value": [
      {
        "firstName": "Bob",
        "lastName":"Smith"
      },
      {
        "firstName": "Sue",
        "lastName":"Baker"
      },
      . . . .
    ]
}
```

File System

Each microservice container will follow Linux's Filesystem Hierarchy Standard and will have a common file structure to simplify everything

High Level directories

Below are the important high-level directories.

Folder	Description	Mappable
`/opt/HeestandTech`	Application binaries and startup scripts	No
`/etc/HeestandTech`	Application configuration files	Yes
`/var/opt/HeestandTech`	Variable data from microservice (temporary upload files, mock data, ect)	Yes
`/var/cache/HeestandTech`	Locally generated data from long running I/O or a calculation. This folder will be able to be deleted with no negative effects to the program	No
`/var/log/HeestandTech`	Log file locations	Yes
`/var/lib/HeestandTech`	Persistent application data	Yes

DevOps Topics

Overview

The purpose of this document is to specify different devops topics from building, testing and deploying to developing in a container with docker.

Visual Studio Code

VS Code will be used for all major development. For the most up to date IDE, it is encouraged that users develop on the cloud development machine. The Remote-Containers and Remote-SSH extensions must be installed. Any additional extensions will be automatically installed when setting up the remote environment

Setup

- On the local machine with VS Code, you will need to generate a keypair for Gitea
  `ssh-keygen -t ed25519 -C "your_email@example.com"`

- Be sure to add it to the agent with
  `eval "$(ssh-agent -s)"`
  `ssh-add <path-to-ssh-key>`

  Once your key has been added to Gitea, you can develop either on a remote machine or your local machine.

On same local machine

- On the local machine with VS Code, generate another keypair to put on the remote machine
  `ssh-keygen -t ed25519 -C "your_email@example.com"`

- On the local machine add the keypair to your ssh agent
  `eval "$(ssh-agent -s)"`
  `ssh-add <path-to-ssh-key>`

- On the remote machine, you can put the **public key** in your ~/.authorized_keys file
- Open the remote location in VS Code

- You should be prompted in the bottom left corner to reopen the project in a container

On a remote machine

- Make sure docker is installed and the docker plugin is installed in VS Code
- Open the folder in VS Code
- You should be prompted in the bottom left corner to reopen the project in a container

Starting a new project

To start a new project, you will need to make a template repository with the Hello World Microservice repository. This can easily be done with Gitea by adding

kheestand/HelloWorldMicroservice as a template repository when creating the repository.

Once you have the HelloWorldMicroservice added as a base repository, you will need to modify the .env file located at /workspace/.openapi/.env. Your .env file should look something like this:

```
{
    "schemaLink" : "<link to openapi schema>"
}
```

Note: you will need a link to a valid openapi schema. You can use the pet store api if you just want to test these instructions.

Once your .env file is ready, you can select the **Start new Project** task in VS code by pressing Ctrl + Shift + P on your keyboard and selecting *Tasks: Run Tasks* and then *Start New Project.*

When the project is generated you should also edit the name of the devcontainer environment to something other than "Microservice v1.0.0.0". This can be done by navigating to the /workspace/.devcontainer/devcontainer.json file and editing the "name" field at the top of the file.

Updating the template repository

In VS code you can use the following commands to update the template

```
git fetch --all
git merge template/master
```

After running git fetch you might need to commit the changes to finish the merge. This can be done through VS Code's Source Control menu. You also might need to allow unrelated histories which you can do by modifying the git merge command to

Viewing Schema in SwaggerUI

The OpenAPI Preview extension should be installed when the devcontainer is set up. If it is, go to the directory /workspace/server/openapi_server/openapi/openapi.yaml then press *Ctrl + Shift + P*. Then select or type in "OpenAPI Preview" from the menu

Run, Test, Build and Deploy

Running the Code for Debugging

Code can be ran in the following ways:

- Press F5
- Go to the Run and Debug menu (Ctrl + Shift + D) and select "Launch OpenAPI2SpringBoot"

Running Tests

In progress...

Building

Jenkins will be used for building all microservices. Each repo will have a dockerfile located in the root directory that will tell the system how to build the image. An example can be found on the HelloMicroservice gitea repo. A sample keystore.jks file is included in the .crypto folder and is used to sign the jar during development. The password for the keystore is test123

Note that microservices that are built in production will be signed using the jarsigner utility with the production keystore.

Deploying

In progress...

What's included in the Dev Container?

Docker-compose environment

MariaDB

Default username *root* and password *mariadb*

Docker Hub Link

VS Code Extensions

VSCode-MySQL-Client

VS Code Store Link

OpenApi-Preview

VS Code Store Link

Security Goals

Overview

The purpose of this document is to outline security goals for the entire system

Goals

There are multiple goals that need to be met to provide users with the most secure platform as possible. We will be defending against many common attacks such as:

- Session hijacking and forging
- Database leaks
- Attackers gain knowledge of the internal system
- Attackers gaining access to the internal system and stealing data

End to End Encryption

All data must be encrypted with some sort of encryption while in transport. The microservices must only accept data from HTTPS sources. Data that is going from the microservice to the database must be encrypted with TLS and data that is at rest or not in use will be encrypted. Messages from one microservice to another will also be encrypted with TLS.

Verifiable Data

Certain data should be signed by the user so other users can verify it was them. Some examples include messages, files, commits, ect.

Authentic Microservices

We should be able to verify microservices before they start that it was signed by a trusted user. Moreover, users should be able to see who has signed the microservice. This problem can be solved by using PGP keys. See the infrastructure page for more information on how it is set up

Cryptic but Useful Errors

Having general errors displayed to the user allows for people to see what went wrong and prevents bad actors from gaining inside knowledge about the system. For example, if a file can not be found we would not display:

```
Traceback (most recent call last):
File "C:\Users\sauris\Desktop\pyfile\folder-read.py", line 13, in <module>
File "C:\Users\sauris\AppData\Local\Programs\Python\Python36-32\lib\bz2.py", l
ine 96, in __init__
self._fp = _builtin_open(filename, mode)
FileNotFoundError: [Errno 2] No such file or directory: 'sampledatafile2.txt.bz2'
```

Instead, we would just display:

```
sampledatafile2.txt.bz2 was not found
```

Non-root Containers

All container/microservices should not be run as the root user. This is to prevent someone from getting access to the container and compromising it.

Post Quantum Encryption

Recently NIST published the chosen algorithms for post quantum cryptography! These will be worked in to the software as seen fit

Infrastructure

Overview

This page will outline the specifications for the infrastructure. It is very important that a strong infrastructure is used to facilitate better scalability, better developer experience and a better user experience. Most infrastructure will remain inaccessible to the general public. The infrastructure will be split up into different "stacks".

All infrastructure containers will be the official images or will be based off of the official docker images

Logging/Metrics

Logging

For logging, the GELF stack will be used (Graylog, elasticsearch, mongodb).

Metrics

For metric capturing the GIT stack will be used (Grafana, InfluxDB, Telegraf)

Identity/Authentication

OpenLDAP and Keycloak will be used for both authentication and authorization. Accounts will be created in OpenLDAP and not Keycloak in most cases

Database

Postgres will be the database of choice for all infrastructure services and application containers. Redis will also be use for caching

File storage

Minio will be used for storing and retrieving files. It is an S3 complaint file system

Cloud Configurations Repository Wiki

How to Contribute

Anyone is welcome to contribute to this project. Be sure to follow the guidelines below and get started!

Folder Structure

The following diagram displays how a developer should set up their API folder structure:

```
APIs/
├─ API_NAME/
│  ├─ master.yaml
│  ├─ models/
│  │  ├─ someModel1.yaml
│  ├─ paths/
│  │  ├─ somePath1.yaml
CommonFragments/
CommonModels/
```

Note that master.yaml contains the info about the API, specific components and the path definitions. An example can be found below:

```
openapi: 3.0.3
info:
  title: "Sample API"
    description: "Information about your API, what it does and how it works"
    version: 0.0.0
servers:
  - url: https://development.service.example.com/
     description: "Development server, breaking changes are often made here"
    - url: https://staging.service.example.com/
      description: "Staging server, breaking changes should rarely happen here"
```

```
    - url: https://service.example.com/
      description: "Production server, breaking changes should never happen here"
paths:
  /path/resource:
    - $ref: ./paths/path-resource.yaml
    /path/something:
    - $ref: ./paths/path-something.yaml
components:
  someComponent ...
security:
  someSecurity ...
tags:
  someTags ...
 externalDocs:
  description: Find more info here
    url: https://docs.example.com/someServiceDoc.html
```

Common Fragments

Common fragments are very small blobs of yaml code that can be shared amongst any services. For example, it is standard that a username has a minimum and maximum length. Therefore, we can create a fragment that we can reuse across all yaml files.

Common Models

Common models are yaml object files that contain code that is shared amongst any services. An example of this is the standard response one might get from the health and version api. Other examples are general HTTP error codes

Logging and Tracing

Purpose

The purpose of this document is to describe how microservices should log. As a note, this excludes infrastructure containers since they all have their own log formats. The goal of having a standard logging format for containers is to allow for easier debugging.

Log Levels

The following standard log levels will be used. A description of each level will also be provided

FATAL	Very sever errors that will lead to the application crashing.
ERROR	Events that could be fatal but also allow the application to continue running.
WARN	Events that could lead to potentially harmful situations.
INFO	Informational messages that highlight the current state of the application
DEBUG	Events that will be useful for a developer when debugging
TRACE	Events that provides finer detail than DEBUG. This could be a dump of a data structure, progress of a task ect.

Standard Logging Format

The standard log format for all services will be a JSON object:

```
{
  "timestamp": "2022-09-22T10:33:20.123456",
  "level": "FATAL|ERROR|WARN|INFO|DEBUG|TRACE",
  "service": "my-microservice",
  "session_id": "7cddeb94-4d9b-4231-bb5a-2c9ce889926c",
  "user_agent": "User agent here",
  "method": {
    "name": "someMethod",
```

```
    "args": ["some", "args"]
  },
  "message": "some super cool message",
    "stacktrace":"the stacktrace here if there is an error"
}
```

PII (Personally Identifiable Information)

Personally Identifiable Information shall be removed from all logs. Some PII includes:

- Username
- Passwords
- API Tokens
- API Keys
- Names
- Age
- Addresses
- Phones
- SSN
- Bank account numbers
- Credit card numbers
- Biometric data
- Geographic data
- Date of birth
- ect

If PII should be put in a log message it should be changed to have only 5 * characters. For instance, the full name "Kyle Heestand" would become "*****". This is to not only protect against the data in the database, but also to protect against people guessing who/what the data is based on the size of it. As an example, take a standard login message:

```
{
  "timestamp": "2022-09-22T10:33:20.123456",
  "level": "INFO",
  "service": "LoginService",
  "session_id": "7cddeb94-4d9b-4231-bb5a-2c9ce889926c",
  "user_agent": "User agent here",
  "method": {
    "name": "login",
    "args": ["kheestand", "some-password"]
  },
  "message": "Kyle Heestand has logged in"
```

```
}
```

After removing the PII, we will get something like this:

```
{
"timestamp": "2022-09-22T10:33:20.123456",
"level": "INFO",
"service": "LoginService",
"session_id": "7cddeb94-4d9b-4231-bb5a-2c9ce889926c",
"user_agent": "User agent here",
"method": {
"name": "login",
"args": ["*****", "*****"]
},
"message": "***** has logged in"
}
```

Obviously, you would more than likely have a different message, but we can see that the user's username, password and full name has been removed to preserve their privacy.

Legal

Legal documents users must agree to before entering the site

Legal

Terms of Service

Terms of Service

Our Terms of Service were last updated on 2/12/2022.

Please read these terms and conditions carefully before using Our Service.

Interpretation and Definitions

Interpretation

The words of which the initial letter is capitalized have meanings defined under the following conditions. The following definitions shall have the same meaning regardless of whether they appear in singular or in plural.

Definitions

For the purposes of these Terms of Service:

- **"Affiliate"** means an entity that controls, is controlled by or is under common control with a party, where "control" means ownership of 50% or more of the shares, equity interest or other securities entitled to vote for election of directors or other managing authority.
- **"Account"** means a unique account created for You to access our Service or parts of our Service.
- **"Company"** (referred to as either "the Company", "We", "Us" or "Our" in this Agreement) refers to Heestand Tech.
- **"Country"** refers to United States of America.
- **"Content"** refers to content such as text, images, or other information that can be posted, uploaded, linked to or otherwise made available by You, regardless of the form of that content.
- **"Device"** means any device that can access the Service such as a computer, a cellphone or a digital tablet.
- **"Feedback"** means feedback, innovations or suggestions sent by You regarding the attributes, performance or features of our Service.
- **"Service"** refers to the Website.
- **"Terms of Service"** (also referred as "Terms" or "Terms of Service") mean these Terms of Service that form the entire agreement between You and the Company regarding the use of the Service.
- **"Third-party Social Media Service"** means any services or content (including data, information, products or services) provided by a third-party that may be displayed, included or made available by the Service.

- **"Website"** refers to Heestand Tech accessible from https://heestand.tech
- **"You"** means the individual accessing or using the Service, or the company, or other legal entity on behalf of which such individual is accessing or using the Service, as applicable.

Acknowledgment

These are the Terms of Service governing the use of this Service and the agreement that operates between You and the Company. These Terms of Service set out the rights and obligations of all users regarding the use of the Service.

Your access to and use of the Service is conditioned on Your acceptance of and compliance with these Terms of Service. These Terms of Service apply to all visitors, users and others who access or use the Service.

By accessing or using the Service You agree to be bound by these Terms of Service. If You disagree with any part of these Terms of Service then You may not access the Service.

You represent that you are over the age of 18. The Company does not permit those under 18 to use the Service without consent from a legal guardian.

Your access to and use of the Service is also conditioned on Your acceptance of and compliance with the Privacy Policy of the Company. Our Privacy Policy describes Our policies and procedures on the collection, use and disclosure of Your personal information when You use the Application or the Website and tells You about Your privacy rights and how the law protects You. Please read Our Privacy Policy carefully before using Our Service.

Your Data

User Accounts

When You create an account with Us, You must provide Us information that is accurate, complete, and current at all times. Failure to do so constitutes a breach of the Terms, which may result in immediate termination of Your account on Our Service.

You are responsible for safeguarding the password that You use to access the Service and for any activities or actions under Your password, whether Your password is with Our Service or a Third-Party Social Media Service.

You agree not to disclose Your password to any third party. You must notify Us immediately upon becoming aware of any breach of security or unauthorized use of Your account.

You may not use as a username the name of another person or entity or that is not lawfully available for use, a name or trademark that is subject to any rights of another person or entity other than You without appropriate authorization, or a name that is otherwise offensive, vulgar or obscene.

How Your Data is Used and Collected

We collect data such as your IP address, browser's user agent, and your browsing data across different sites to determine legitimate users and how the site is being used. You may opt out of sharing your user agent and browsing data by submitting a DNT header with your HTTP requests.

To learn more about the DNT (Do Not Track) header, you can visit allaboutdnt.com.

We will never sell your personal or usage data however by accepting these Terms of Service You give the Company permission to run anonymized studies including but not limited to how the site is used.

Content

Your Right to Post Content

Our Service allows You to post Content. You are responsible for the Content that You post to the Service, including its legality, reliability, and appropriateness.

By posting Content to the Service, You grant Us the right and license to publicly display such Content on and through the Service. You do retain any and all of Your rights to any Content You submit, post or display on or through the Service and You are responsible for protecting those rights. You agree that this license includes the right for Us to make Your Content available to other users of the Service, who may also use Your Content subject to these Terms.

You represent and warrant that: (i) the Content is Yours (You own it) or You have the right to use it and grant Us the rights and license as provided in these Terms, and (ii) the posting of Your Content on or through the Service does not violate the privacy rights, publicity rights, copyrights, contract rights or any other rights of any person.

Content Restrictions

The Company is not responsible for the content of the Service's users. You expressly understand and agree that You are solely responsible for the Content and for all activity that occurs under your account, whether done so by You or any third person using Your account.

You may not transmit any Content that is unlawful, offensive, upsetting, intended to disgust, threatening, libelous, defamatory, obscene or otherwise objectionable. Examples of such objectionable Content include, but are not limited to, the following:

- Unlawful or promoting unlawful activity.
- Defamatory or discriminatory content, including references or commentary about religion, race, sexual orientation, gender, national/ethnic origin, or other targeted groups.
- Spam, machine – or randomly – generated, constituting unauthorized or unsolicited advertising, chain letters, any other form of unauthorized solicitation, or any form of lottery or gambling.
- Containing or installing any viruses, worms, malware, trojan horses, or other content that is designed or intended to disrupt, damage, or limit the functioning of any software, hardware or telecommunications equipment or to damage or obtain unauthorized access to any data or other information of a third person.

- Infringing on any proprietary rights of any party, including patent, trademark, trade secret, copyright, right of publicity or other rights.
- Impersonating any person or entity including the Company and its employees or representatives.
- Violating the privacy of any person.
- False information and features.
- Information intended to enrage other users or targeted groups

The Company reserves the right, but not the obligation, to, in its sole discretion, determine whether or not any Content is appropriate and complies with this Terms, refuse or remove this Content. The Company further reserves the right to make formatting and edits and change the manner of any Content. The Company can also limit or revoke the use of the Service if You post such objectionable Content.
As the Company cannot control all content posted by users and/or third parties on the Service, you agree to use the Service at your own risk. You understand that by using the Service You may be exposed to content that You may find offensive, indecent, incorrect or objectionable, and You agree that under no circumstances will the Company be liable in any way for any content, including any errors or omissions in any content, or any loss or damage of any kind incurred as a result of your use of any content.

Content Backups

Although regular backups of Content are performed, the Company does not guarantee there will be no loss or corruption of data.

Corrupt or invalid backup points may be caused by, without limitation, Content that is corrupted prior to being backed up or that changes during the time a backup is performed.

The Company will provide support and attempt to troubleshoot any known or discovered issues that may affect the backups of Content. But You acknowledge that the Company has no liability related to the integrity of Content or the failure to successfully restore Content to a usable state.

You agree to maintain a complete and accurate copy of any Content in a location independent of the Service.

Copyright Policy

Intellectual Property Infringement

We respect the intellectual property rights of others. It is Our policy to respond to any claim that Content posted on the Service infringes a copyright or other intellectual property infringement of any person.

If You are a copyright owner, or authorized on behalf of one, and You believe that the copyrighted work has been copied in a way that constitutes copyright infringement that is taking place through the Service, You must submit Your notice in writing to the attention of our copyright agent via email (dmca@heestand.tech) and include in Your notice a detailed description of the alleged

infringement.

You will be held accountable for damages (including costs and attorneys' fees) for misrepresenting that any Content is infringing Your copyright.

DMCA Notice and DMCA Procedure for Copyright Infringement Claims

You may submit a notification pursuant to the Digital Millennium Copyright Act (DMCA) by providing our Copyright Agent with the following information in writing (see 17 U.S.C 512(c)(3) for further detail):

- An electronic or physical signature of the person authorized to act on behalf of the owner of the copyright's interest.
- A description of the copyrighted work that You claim has been infringed, including the URL (i.e., web page address) of the location where the copyrighted work exists or a copy of the copyrighted work.
- Identification of the URL or other specific location on the Service where the material that You claim is infringing is located.
- Your address, telephone number, and email address.
- A statement by You that You have a good faith belief that the disputed use is not authorized by the copyright owner, its agent, or the law.
- A statement by You, made under penalty of perjury, that the above information in Your notice is accurate and that You are the copyright owner or authorized to act on the copyright owner's behalf.

You can contact our copyright agent via email (dmca@heestand.tech). Upon receipt of a notification, the Company will take whatever action, in its sole discretion, it deems appropriate, including removal of the challenged content from the Service. If it is deemed by Us that You, a person or a person on behalf of the owner of the copyright interest is purposefully submitting claims in bad faith, the requests will be ignored, and no action will be taken.

Intellectual Property

The Service and its original content (excluding Content provided by You or other users), features and functionality are and will remain the exclusive property of the Company and its licensors.

The Service is protected by copyright, trademark, and other laws of both the Country and foreign countries.

Our trademarks and trade dress may not be used in connection with any product or service without the prior written consent of the Company.

Your Feedback to Us

You assign all rights, title and interest in any Feedback You provide the Company. If for any reason such assignment is ineffective, You agree to grant the Company a non-exclusive, perpetual, irrevocable, royalty free, worldwide right and license to use, reproduce, disclose, sub-license,

distribute, modify and exploit such Feedback without restriction.

Links to Other Websites

Our Service may contain links to third-party web sites or services that are not owned or controlled by the Company.

The Company has no control over, and assumes no responsibility for, the content, privacy policies, or practices of any third party web sites or services. You further acknowledge and agree that the Company shall not be responsible or liable, directly or indirectly, for any damage or loss caused or alleged to be caused by or in connection with the use of or reliance on any such content, goods or services available on or through any such web sites or services.

We strongly advise You to read the terms and conditions and privacy policies of any third-party web sites or services that You visit.

Termination

We may suspend Your Account immediately, without prior notice or liability, for any reason whatsoever, including without limitation if You breach these Terms of Service. After suspension, you will be subject to meet with Our team to discuss why your account was suspended. If you fail to meet with Our team and resolve the issue, your account will be terminated. We will NEVER terminate an account without human review. Accounts that have been suspended for more than 90 days are subject to automatic removal.

Upon termination, Your right to use the Service will cease immediately. If You wish to terminate Your Account, You may request Your account to be deleted.

Limitation of Liability

Notwithstanding any damages that You might incur, the entire liability of the Company and any of its suppliers under any provision of this Terms and Your exclusive remedy for all of the foregoing shall be limited to the amount actually paid by You through the Service or 0.69 USD if You haven't purchased anything through the Service.

To the maximum extent permitted by applicable law, in no event shall the Company or its suppliers be liable for any special, incidental, indirect, or consequential damages whatsoever (including, but not limited to, damages for loss of profits, loss of data or other information, for business interruption, for personal injury, loss of privacy arising out of or in any way related to the use of or inability to use the Service, third-party software and/or third-party hardware used with the Service, or otherwise in connection with any provision of this Terms), even if the Company or any supplier has been advised of the possibility of such damages and even if the remedy fails of its essential purpose.

Some states do not allow the exclusion of implied warranties or limitation of liability for incidental or consequential damages, which means that some of the above limitations may not apply. In these states, each party's liability will be limited to the greatest extent permitted by law.

"AS IS" and "AS AVAILABLE" Disclaimer

The Service is provided to You "AS IS" and "AS AVAILABLE" and with all faults and defects without warranty of any kind. To the maximum extent permitted under applicable law, the Company, on its own behalf and on behalf of its Affiliates and its and their respective licensors and service providers, expressly disclaims all warranties, whether express, implied, statutory or otherwise, with respect to the Service, including all implied warranties of merchantability, fitness for a particular purpose, title and non-infringement, and warranties that may arise out of course of dealing, course of performance, usage or trade practice. Without limitation to the foregoing, the Company provides no warranty or undertaking, and makes no representation of any kind that the Service will meet Your requirements, achieve any intended results, be compatible or work with any other software, applications, systems or services, operate without interruption, meet any performance or reliability standards or be error free or that any errors or defects can or will be corrected.

Without limiting the foregoing, neither the Company nor any of the company's provider makes any representation or warranty of any kind, express or implied: (i) as to the operation or availability of the Service, or the information, content, and materials or products included thereon; (ii) that the Service will be uninterrupted or error-free; (iii) as to the accuracy, reliability, or currency of any information or content provided through the Service; or (iv) that the Service, its servers, the content, or e-mails sent from or on behalf of the Company are free of viruses, scripts, trojan horses, worms, malware, timebombs or other harmful components.

Some jurisdictions do not allow the exclusion of certain types of warranties or limitations on applicable statutory rights of a consumer, so some or all of the above exclusions and limitations may not apply to You. But in such a case the exclusions and limitations set forth in this section shall be applied to the greatest extent enforceable under applicable law.

Governing Law

The laws of the Country, excluding its conflicts of law rules, shall govern this Terms and Your use of the Service. Your use of the Application may also be subject to other local, state, national, or international laws.

Waiver of Right to Sue

Except for the Company’s promises and obligations contained in this agreement, You further agree, promise and covenant that neither You, nor any person, organization, or any other entity acting on your behalf will file, charge, claim, sue or cause or permit to be filed, charged or claimed, any action for damages or other relief (including injunctive, declaratory, monetary relief or other) against the Company and any affiliates, involving any matter occurring in the past up to the Effective Date of this Agreement or involving any continuing effects of actions or practices which arose prior to the Effective Date of this Agreement.

Disputes Resolution

If You have any concern or dispute about the Service, You agree to resolve the dispute informally by contacting the Company. If We can not resolve the dispute with You within reason, no further

action will be taken.

For European Union (EU) Users

If You are a European Union consumer, you will benefit from any mandatory provisions of the law of the country in which you are resident in.

United States Legal Compliance

You represent and warrant that (i) You are not located in a country that is subject to the United States government embargo, or that has been designated by the United States government as a "terrorist supporting" country, and (ii) You are not listed on any United States government list of prohibited or restricted parties.

Severability and Waiver

Severability

If any provision of these Terms is held to be unenforceable or invalid, such provision will be changed and interpreted to accomplish the objectives of such provision to the greatest extent possible under applicable law and the remaining provisions will continue in full force and effect.

Waiver

Except as provided herein, the failure to exercise a right or to require performance of an obligation under these Terms shall not affect a party's ability to exercise such right or require such performance at any time thereafter nor shall be the waiver of a breach constitute a waiver of any subsequent breach.

Translation Interpretation

These Terms of Service may have been translated if We have made them available to You on our Service.
You agree that the original English text shall prevail in the case of a dispute.

Changes to These Terms of Service

We reserve the right, at Our sole discretion, to modify or replace these Terms at any time. If a revision is material We will make efforts to provide at least 30 days' notice prior to any new terms taking effect. What constitutes a material change will be determined at Our sole discretion.

By continuing to access or use Our Service after those revisions become effective, You agree to be bound by the revised terms. If You do not agree to the new terms, in whole or in part, please stop using the website and the Service.

Contact Us

If you have any questions about these Terms of Service, You can contact us by sending us an email to contact@heestand.tech

Virtual Machine Infrastructure

Overall Goal

The overall goal of this project is to be able to maintain virtual machines easier and provide a secure and stable environment for docker-compose stacks and other binaries to run. One requirement is virtual machines should be generated on demand to allow for scalability. This could be achieved by netbooting a Linux operating system, configuring the machine with cloud-init and then deploying a docker-compose stack on it.

> Note that OPNSense and the WIndows Machine are not discussed in this document. While they do exist, they will not be a part of this infrastructure

Final Solution

Generating VM's on Demand

Using the power of the Debian Live project, we are able to customize a live netboot image to provide a base operating system for all VMs. This will also allow us to generate "live cds" on demand and then push the update to a central boot server where the VMs can then boot the latest copy with the latest kernel and bug fixes. See Gitea repo here

Using Cloud-Init

The Xen Orchestra appliance has built in support for using a cloud-init template. Users will be able to specify whichever packages they want installed and other configuration items. Docker compose will be deployed using cloud-init

Ideal Implementation

The Genesis Machine

Function

This machine will provide all other Virtual Machines with the latest image to boot up via PXE. Additionally, this machine will house its own version of Gitea and Jenkins. This machine will also be home to LDAP and Keycloak for identity management since they have both been identified as critical services.

VM Resources

- 2 CPU Cores
- 2 GB of RAM

Networking

- Internal Hostname: genesis.heestand.local
- IP: 192.168.2.10

Containers Used

- Gitea
- Jenkins
- Netbootxyz
- Traefik
- SmallStep
- LDAP
- FusionDirectory
- Keycloak
- Portainer

Other Configurations

- Docker Engine v2
- Encrypted overlay network

Edge Machine

Function

This machine will serve as the edge for incoming traffic. All incoming traffic to all services must go through here

VM Resources

- 1 CPU Cores
- 2 GB of RAM

Networking

- Internal Hostname: edge.heestand.local
- IP: 192.168.2.11

Containers Used

- Traefik
- Fail2ban?

Other Configurations

- Docker Engine v2
- smallstep client (system)
- portainer agent (auto startup docker container)
- Custom SSH shell to portal to different machines?
- Encrypted overlay network

Database Machine

Function

This machine will provide all of the databases for the services machine.

VM Resources

- 8 CPU Cores
- 8 GB of RAM

Networking

- Internal Hostname: database.heestand.local
- IP: 192.168.2.12

Containers Used

- MariaDB
- MongoDB
- Postgres
- Redis
- MySQL
- InfluxDB
- mindmax
- elasticsearch
- Adminer

Other Configurations

- Docker Engine v2
- smallstep client (system)
- portainer agent (auto startup docker container)
- Encrypted overlay network

Prod Service Machine

Function

This machine will provide all of the services

VM Resources

- 2 CPU Cores
- 8 GB of RAM

Networking

- Internal Hostname: prod.heestand.local
- IP: 192.168.2.13

Containers Used

- Traefik
- <Insert other services here>

Other Configurations

- Docker Engine v2
- smallstep client
- Encrypted overlay network

Dev Service Machine

Function

This machine will provide development versions of services

VM Resources

- 1 CPU Cores
- 8 GB of RAM

Networking

- Internal Hostname: dev.heestand.local
- IP: 192.168.2.14

Containers Used

- Traefik
- <Insert other services here>

Other Configurations

- Docker Engine v2
- smallstep client (system)
- portainer agent (auto startup docker container)
- Encrypted overlay network

Logging and Metrics Machine

Function

This machine will be used for logging and managing the virtual machines

VM Resources

- 1 CPU Cores
- 4 GB of RAM

Networking

- Internal Hostname: dev.heestand.local
- IP: 192.168.2.15

Containers Used

- traefik
- grafana
- graylog

Other Configurations

- Docker Engine v2
- smallstep client (system)
- portainer agent (auto startup docker container)

Build Machine

Function

This machine will replace Linuxbox and will be used for development activities. It will also host the big Jenkins build cluster.

VM Resources

- 20 CPU Cores
- 64 GB of RAM

Networking

- Internal Hostname: linuxbox.heestand.local
- IP: 192.168.2.16

Containers Used

Other Configurations

- Jenkins installed directly

Implementation Notes

Operating System Security

Using a live CD with a readonly file system can be tricky, below are the current security precautions that are being taken:

User and Permission Changes

- User "live" (default live user) will have it's password locked to prevent ssh login
- User "live" will be assigned the shell /usr/sbin/nologin to prevent anyone from logging in as that user
- A user will not be allowed to ssh into the system as user "root"
- Console will be disabled to prevent rogue admins and unauthorized access

> Due to the above security measures, you MUST provide a SSH key to the Virtual Machine for the user debian (or whatever user you made via the cloud-init process). If you do not, you

will not be able to remote into the machine!

File System

The file system for the operating system is not encrypted and will be lost on reboot, it is not recommended you store non-critical information on it. Use your dedicated data disk to store information instead.