Hosting Platform & API Gateway

Since Bristlecone microservices are already running at Amazon it would be just fine to take advantage of that and use some technology like the one they offer for micro services:

* *AWS Beanstalk*
* *Potentially AWS API Gateway* ***(we need to evaluate and try that since we anyways will need to create an API Gateway for public API)***

# AWS Beanstalk

It is a very good technology from Amazon we can use to create micro services since it support a great number of technologies (making our infrastructure technology agnostic) including .NET Core which is the main requirement we have for Bristlecone Financing application. If in the future a new kind of service (let’s say *nodejs*) is required that could be done.

Supported Technology:

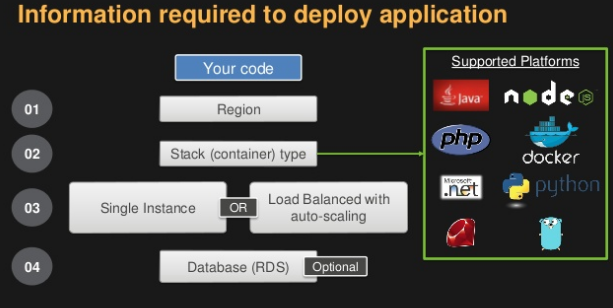
* *Java*
* *.NET*
* *PHP*
* *NodeJS*
* *Python*
* *Ruby*
* *Go*

Supported Servers:

* *Docker*
* *Apache*
* *Nginx*
* *Passenger*
* *IIS*

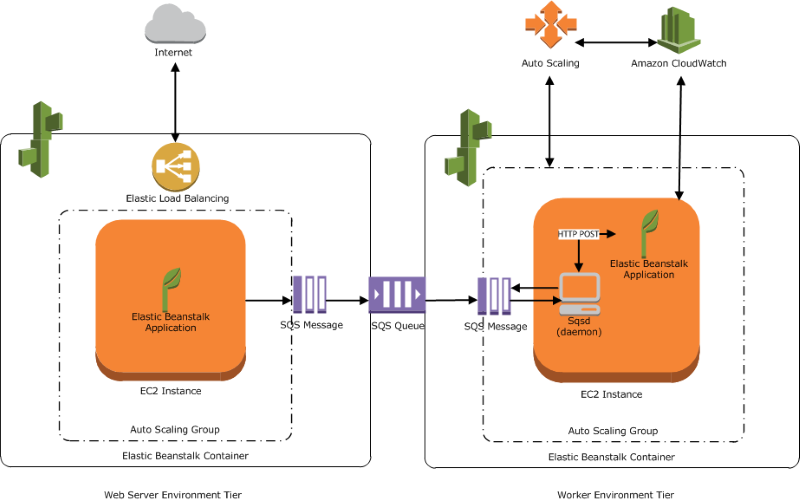
Cross-Cutting Concerns supported by Elastic Beanstalk:

* *Auto-Scaling*
* *Monitoring*
* *Load Balancing*



Potentially Elastic Beanstalk micro services could be placed into a *VPC (Virtual Private Cloud).* ***We would need to test if possible to access from a public API to those resources so in that way we only allow the API Gateway to be public.***

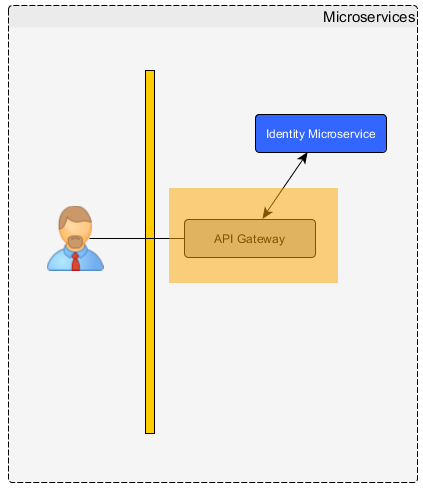
# Amazon Beanstalk Architecture Overview



# API Gateway Architecture

The API Gateway is a micro service with a public HTTP API that covers all the system functionality. The API Gateway acts like an adapter between application and Micro services.

The API Gateway is in charge of handle incoming requests and security concerns in collaboration with a Login micro service which preferable will be isolated as well.



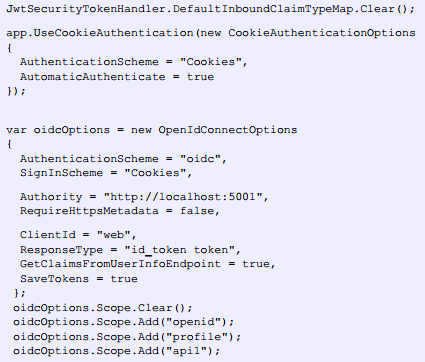
That API Gateway is in charge of login the user using the application by:

* *Cookie*
* *Jwt*

It mainly could use a nugget package called ***OpenIdConnect*** which allows clients of all types to request and receive information about authenticated session and end-users performed by an Authorization Server (*aka authority*). It has support for .NET Core.

<http://openid.net/connect/>

Below is an example code to create cookies against an Authorization Server using OpenIdConnect which will generate a session cookie to the web application as an example:



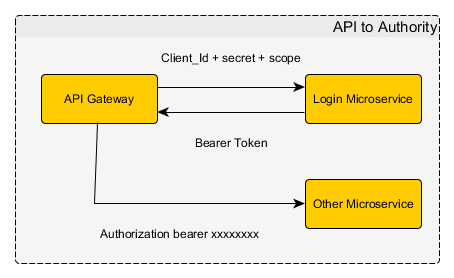
The collaboration with other services and the gateway module will look like the sample below:



As we use clients the application will present a small temporal coupling with other services. (A way to remove that is using a service bus if wished).

# Communicating API Gateway – Micro services System

In order to communicate to other micro services using client classes for doing it, API Gateway has to request access to the Identity Micro service which will return a JWT (*Json web Token*) if valid. (Since probably the API Gateway become the last thing to do that would not possible at the beginning).

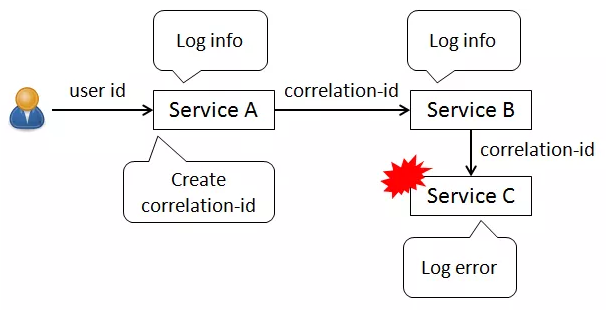


***An HttpClientFactory component would be necessary for adding common headers and requesting a JWT before sending out the request. That could be part of a nugget package.***



# Correlation ID

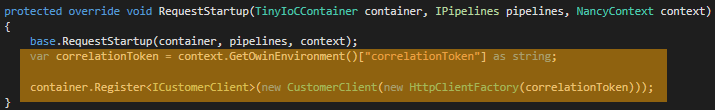
It is an ID value to uniquely identify an entity across multiple systems. In that case the correlation token usually will be propagated from the API Gateway through all micro services involved in the request.



That’s the example when a micro service is broken. Without that we should not be able to track the request from the beginning. So every incoming request will attach a correlation id to the outgoing request into a header. Then a middleware (which could be a common library in out .NET Micro services Platform) check for the incoming request verifying the presence of the ***Correlation Id*** as well as doing logging.

A way to generate correlations id could be using DI to inject the correlation id to the components making requests and an Owin middleware.

***Example:*** Injecting the ***CorrelationId*** (created by an ***OWIN*** middleware) to an ***HttpClientFactory*** class.



***Example:*** Use a middleware to generate the correlation id.

