COMP7940 Group Project

Milestone-4

1. How is your project architecture related to the theory taught in the lecture?

   Using the three-tier architecture, there is a one-to-one mapping between logical and physical elements. The first layer is the physical layer, the device that runs the line-bot, the mobile phone. The second layer is the application services layer, or line-bot code deployment layer. The third layer is the database layer, corresponding to the redis database in our project.

   It also uses the National Institute of Standards and Technology (NIST) architecture. In a simple word, this is an outsourcing. Cloud model heroku and crawler technology in our project are external resources in the architecture.

   In addition, with the RMI architecture, the remote machine is our github, which is where our code is managed, and the code is on the local machine, so there is some interaction between the two.

   Then there is the SOA architecture. Using the restful API in it, the POST function is defined in the network protocol. The method in the app_route() function in the project uses post. And our application usage process also obeys this architecture. The user sends the message to the line bot account, and the line bot receives the message and posts it to the Webhook URL using the https protocol. The Webhook URL is what we call the web service, which is responsible for the actual processing of the message.

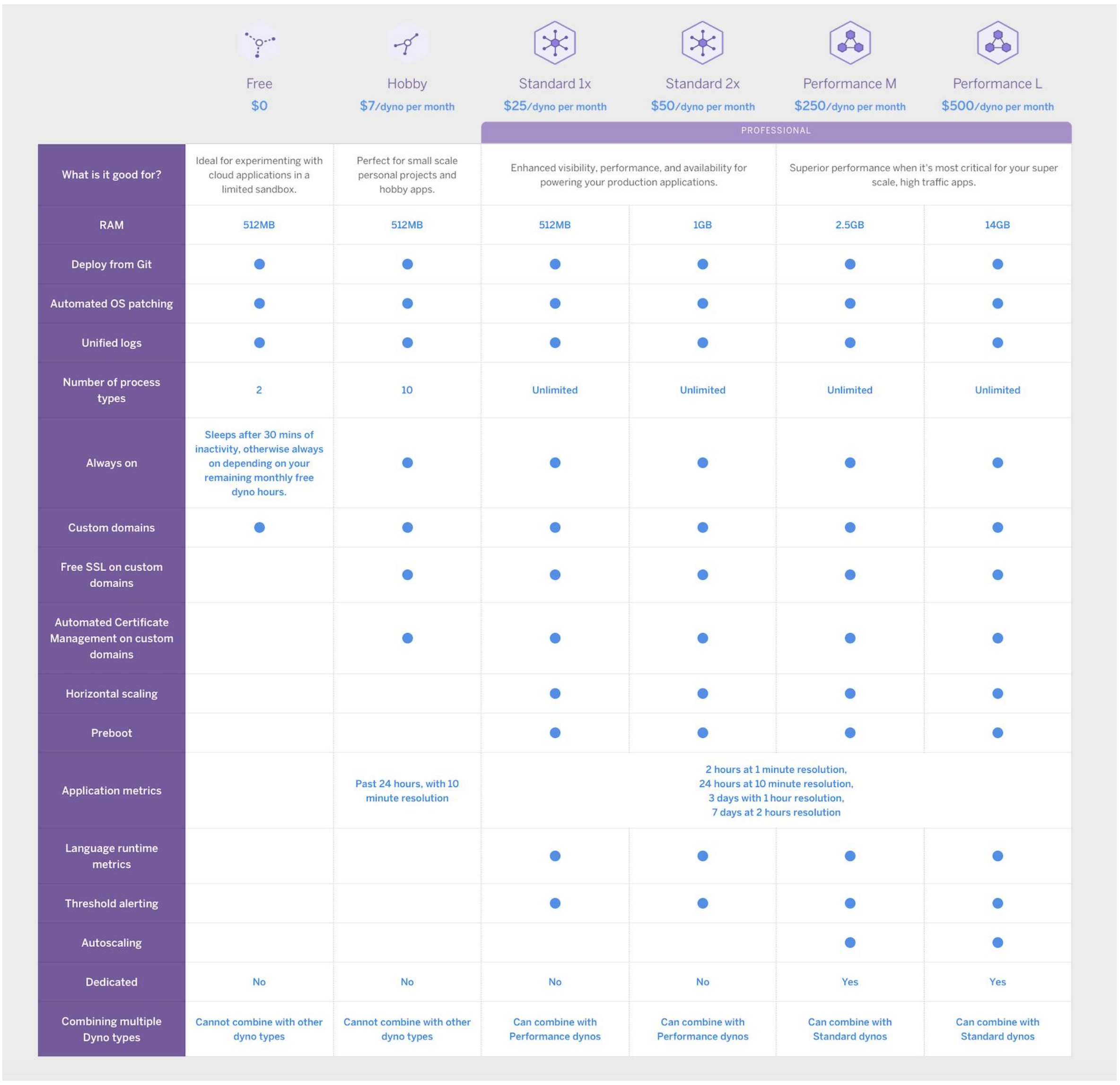
   The overall architecture is shown in picture-1.

2. Can you demonstrate, with some screen cap, how to increase capacity of your chat bot service?

   Since our chatbot is deployed on the Heroku platform, we can change the type of dyno to increase capacity of our chatbot service. Dynos are the heart of the Heroku platform. The Heroku platform use the container model to run and scale all Heroku applications. The container used in Heroku is called "dynos". Chatbot can scale to any specified number of dynos based on its resource demands. Heroku platform offers many types of dyno to support all size applications. Memory, CPU share, and other differentiating characteristics for each dyno type are listed below:

Dyno Type	Memory (RAM)	CPU Share	Compute	Dedicated	Sleeps
free	512 MB	1x	1x-4x	no	yes
hobby	512 MB	1x	1x-4x	no	no
standard-1x	512 MB	1x	1x-4x	no	no
standard-2x	1024 MB	2x	4x-8x	no	no
performance-m	2.5 GB	100%	11x	yes	no
performance-l	14 GB	100%	46x	yes	no

We can choose the appropriate dynos according to the usage of our chatbot service. But depending on the configuration, different dynos will charge different prices:



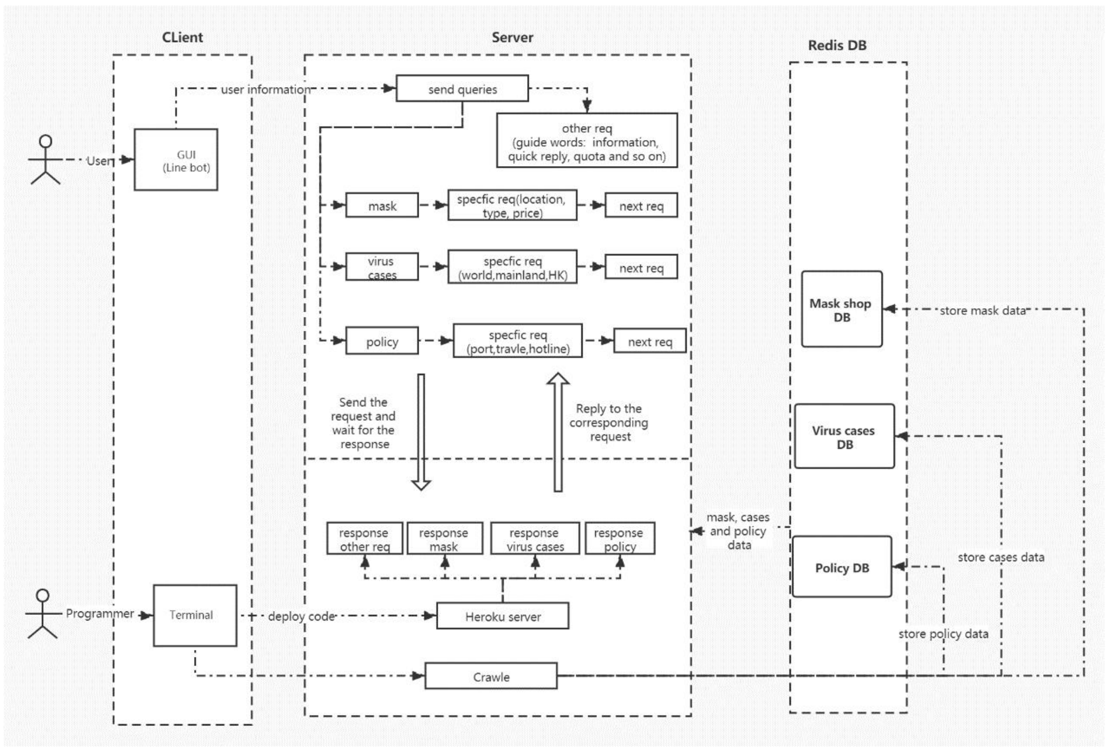
	Free $0	Hobby $7/dyno per month	Standard 1x $25/dyno per month	Standard 2x $50/dyno per month	Performance M $250/dyno per month	Performance L $500/dyno per month
			PROFESSIONAL			
What is it good for?	Ideal for experimenting with cloud applications in a limited sandbox.	Perfect for small scale personal projects and hobby apps.	Enhanced visibility, performance, and availability for powering your production applications.		Superior performance when it's most critical for your super scale, high traffic apps.	
RAM	512MB	512MB	512MB	1GB	2.5GB	14GB
Deploy from Git	●	●	●	●	●	●
Automated OS patching	●	●	●	●	●	●
Unified logs	●	●	●	●	●	●
Number of process types	2	10	Unlimited	Unlimited	Unlimited	Unlimited
Always on	Sleeps after 30 mins of inactivity, otherwise always on depending on your remaining monthly free dyno hours.	●	●	●	●	●
Custom domains	●	●	●	●	●	●
Free SSL on custom domains		●	●	●	●	●
Automated Certificate Management on custom domains		●	●	●	●	●
Horizontal scaling			●	●	●	●
Preboot			●	●	●	●
Application metrics		Past 24 hours, with 10 minute resolution	2 hours at 1 minute resolution, 24 hours at 10 minute resolution, 3 days with 1 hour resolution, 7 days at 2 hours resolution			
Language runtime metrics			●	●	●	●
Threshold alerting			●	●	●	●
Autoscaling					●	●
Dedicated	No	No	No	No	Yes	Yes
Combining multiple Dyno types	Cannot combine with other dyno types	Cannot combine with other dyno types	Can combine with Performance dynos	Can combine with Performance dynos	Can combine with Standard dynos	Can combine with Standard dynos

In addition to different configurations and prices, different types of dynos offer different services. We can choose the most appropriate dynos according to our needs and financial resources to meet the capacity requirements of our chatbot service.

Besides, because most of our chatbot data is stored in Redis. We can also optimize Redis to increase our capacity of our chatbot service. For Redis, we can use Resque or Sidekiq to implement multithreading. Sidekiq is an open source job scheduler written in Ruby. It uses Redis as an in-memory data structure store and is written in Ruby. It can be used with Resque, another Redis based job scheduler, or more commonly as a standalone product. Sidekiq reads jobs from a Redis queue, using the First In First Out (FIFO) model, to process jobs. Job processing is asynchronous and allows a web thread serve requests, rather than process heavy tasks. Therefore, we can process requests from multiple users to get data from Redis at the same time. It is a good way to increase our capacity of our chatbot service.

3. Can you identify if you bot is one of the example of PaaS, IaaS, SaaS? Explain your answer.

   Our bot is PaaS. In this project, we used heroku to implement it. When we did this project, we just can write the line bot code, operate the data by redis and configure the configurable parameters of the environment that the app hosts. Meanwhile, it was unnecessary to care about the networking, storage, servers, O/S and other infrastructures. All of them were managed by the heroku. Therefore, the process satisfied the definition of PaaS.



Picture -1 line-bot overall architecture

Team members:

ZHOU Junhao
GUO Zijia
LIU liu