Hosting Microservices on DigitalOcean

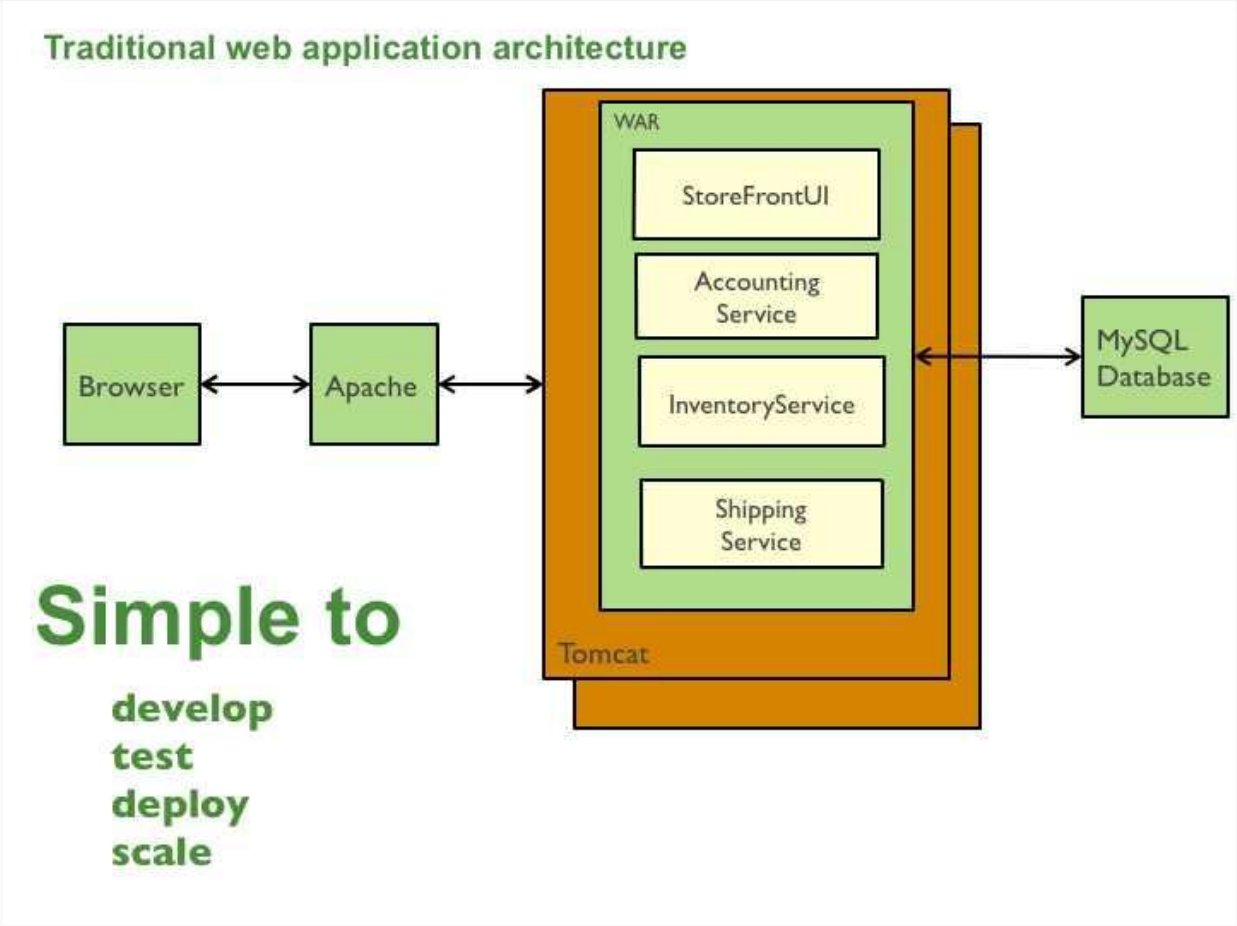
DigitalOcean Hyderabad Second Meetup - July 2016

uday.jandhyala@caribou tech.com

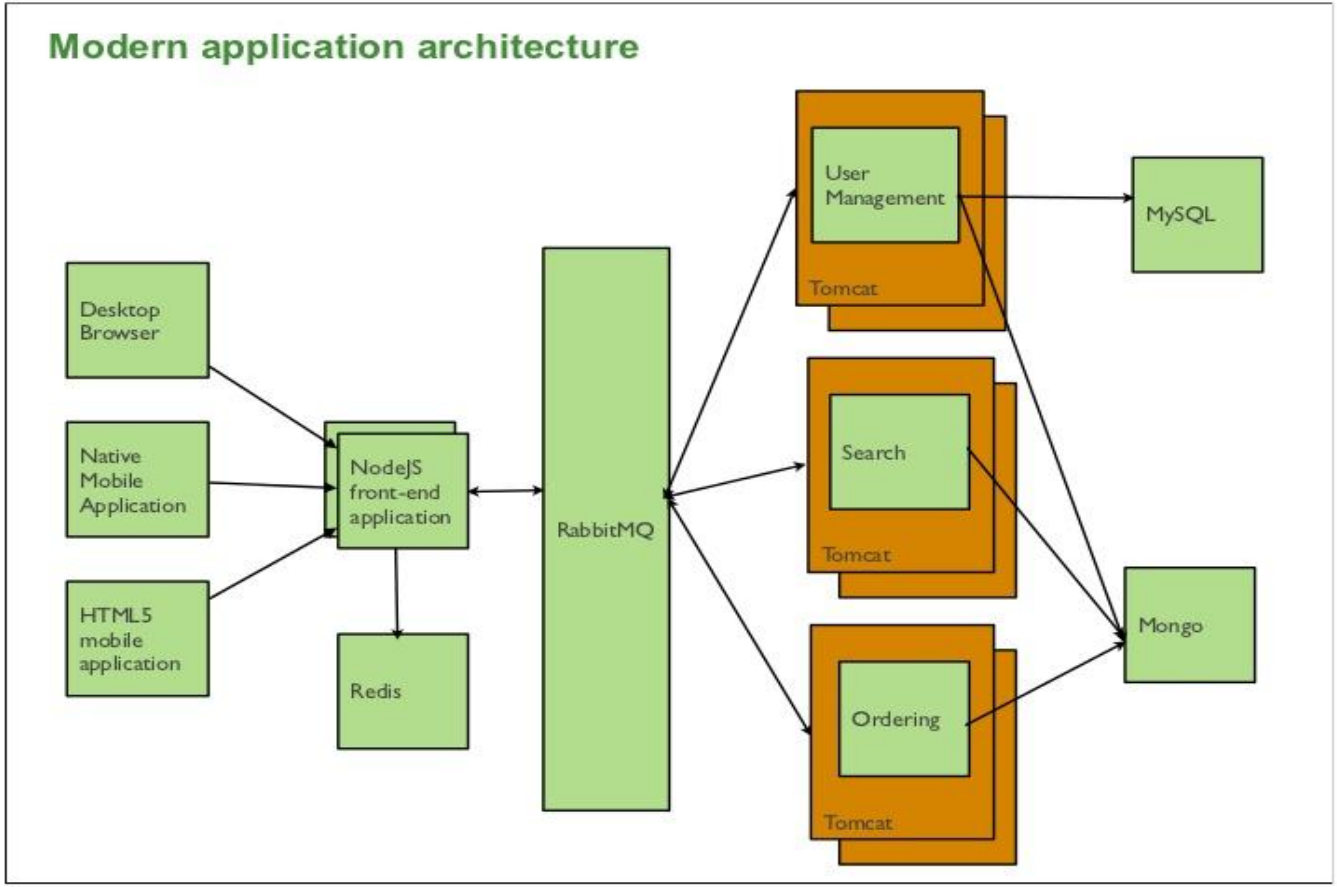
Agenda

1. Why Microservices
2. How to set up Microservices infrastructure
3. Hosting all Microservices on a single node
4. Hosting Microservices on multiple nodes

Monolithic Web Application Design



Modern Web Application Design



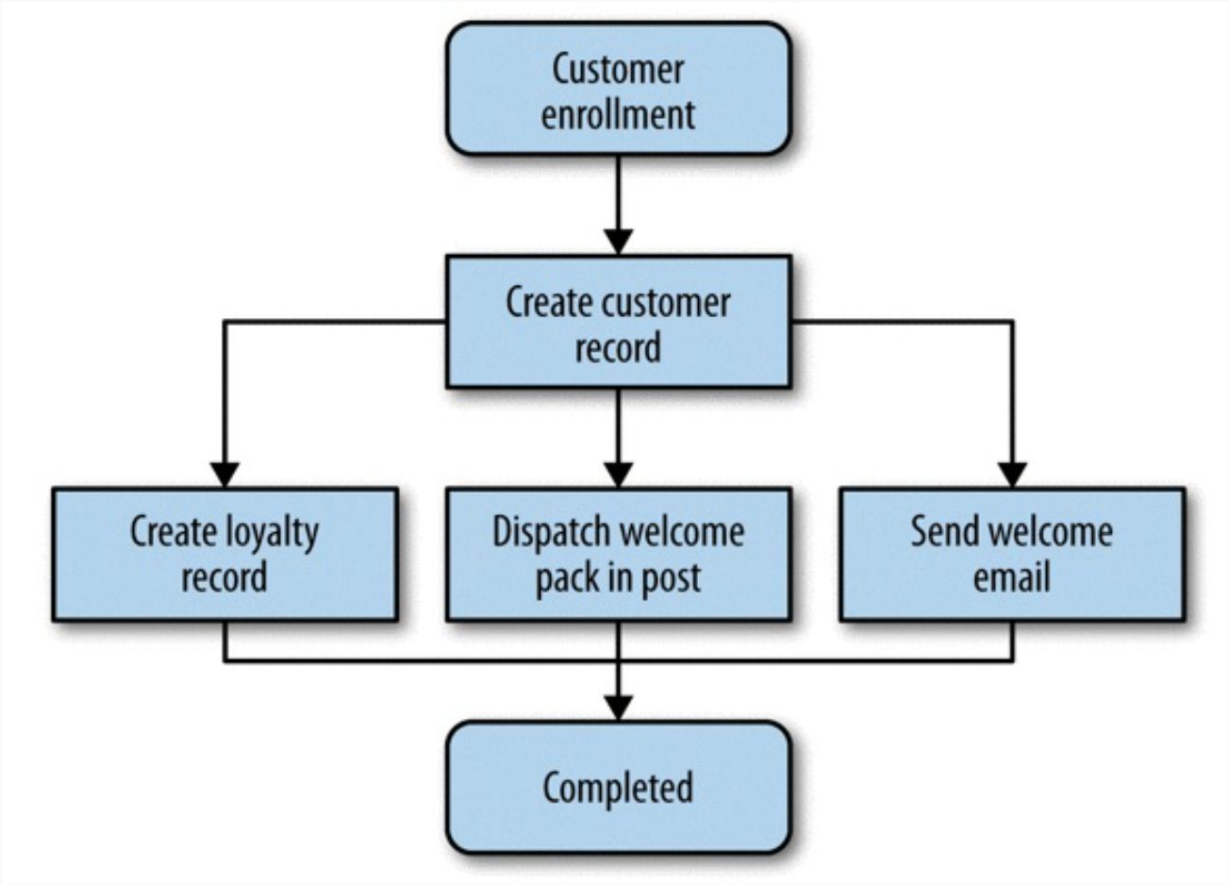
Two modes of communication in Web Applications

Request / Response

&

Event based

Orchestration vs Choreography (Real life scenario)



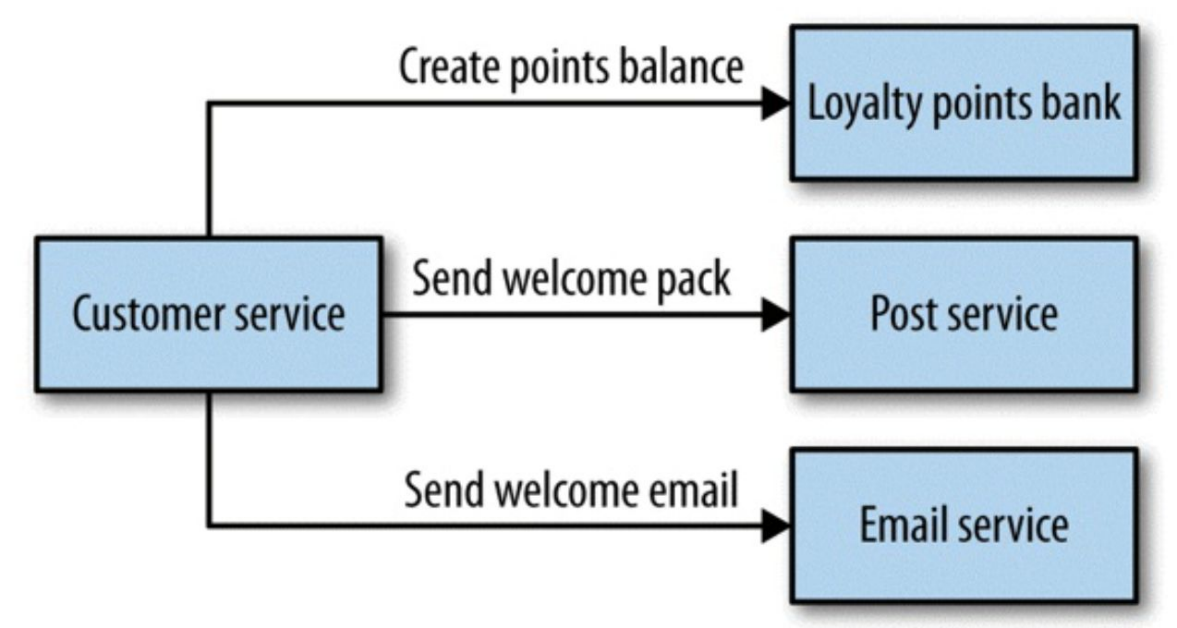
When a new Customer joins

Let’s take a small example of when a new Customer joins a Cloud Service

1. Create a new record for the new User in the database Table(s)
2. Postal system to send out a welcome pack
3. Send a welcome email to the Customer

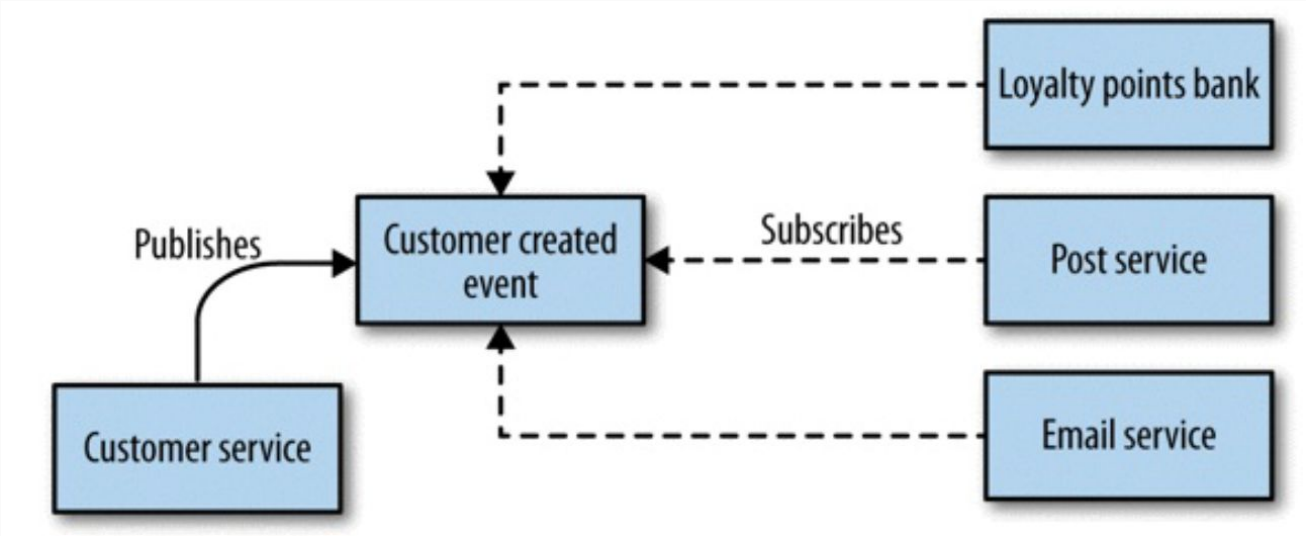
Orchestration style of design

- A central brain (**CustomerService**), that drives the process, much like the conductor in an Orchestra
- Downside of this approach is that ‘CustomerService’ could become too much of a **governing authority where logic starts to grow** as features add up

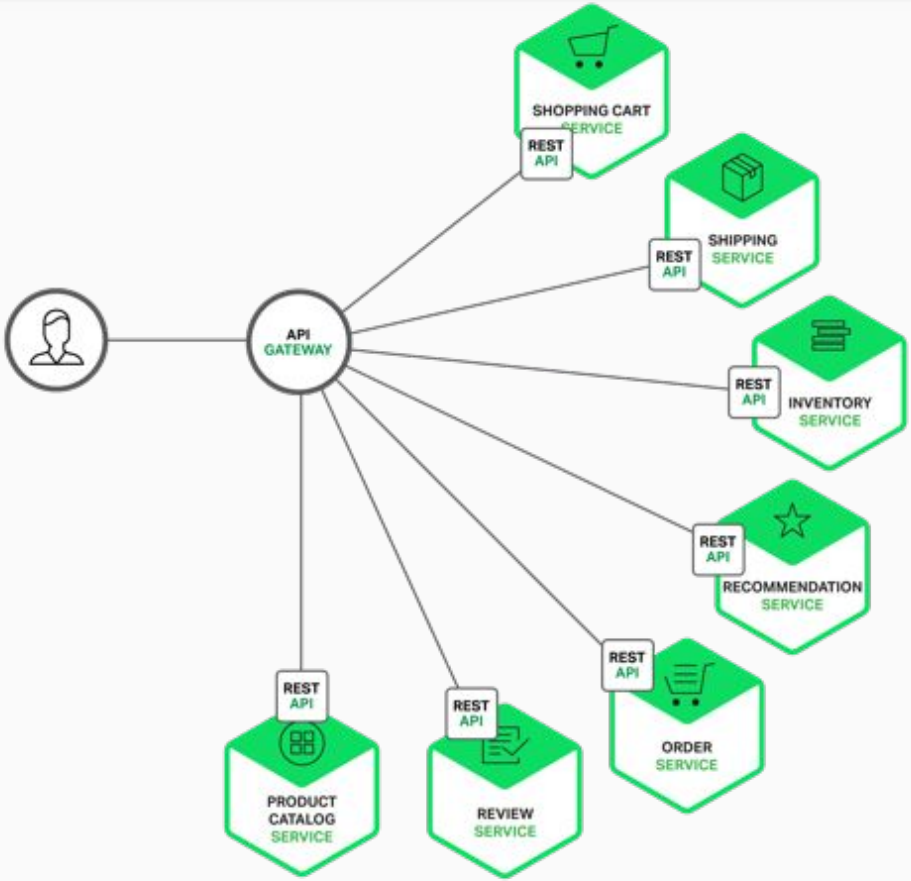


Choreography style of design

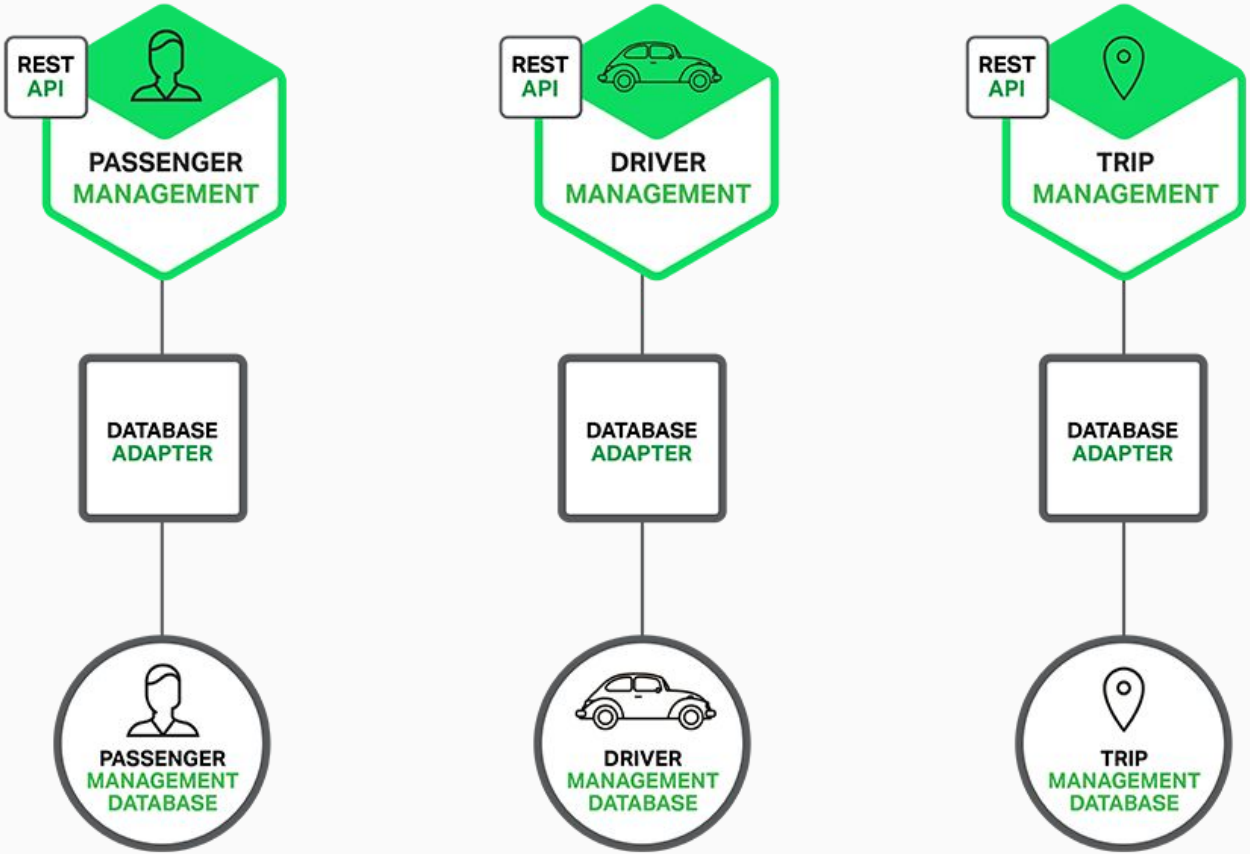
- **‘CustomerService’** informs each part of the system of its job, and let it work out the details, like all dancers finding their way and reacting to others around them in a ballet
- CustomerService emits a **‘customer_created’** event, to which other components ‘subscribe’ and react accordingly
- **Event driven systems** are much more loosely coupled



Microservices Architecture



Anotomy of each Microservice



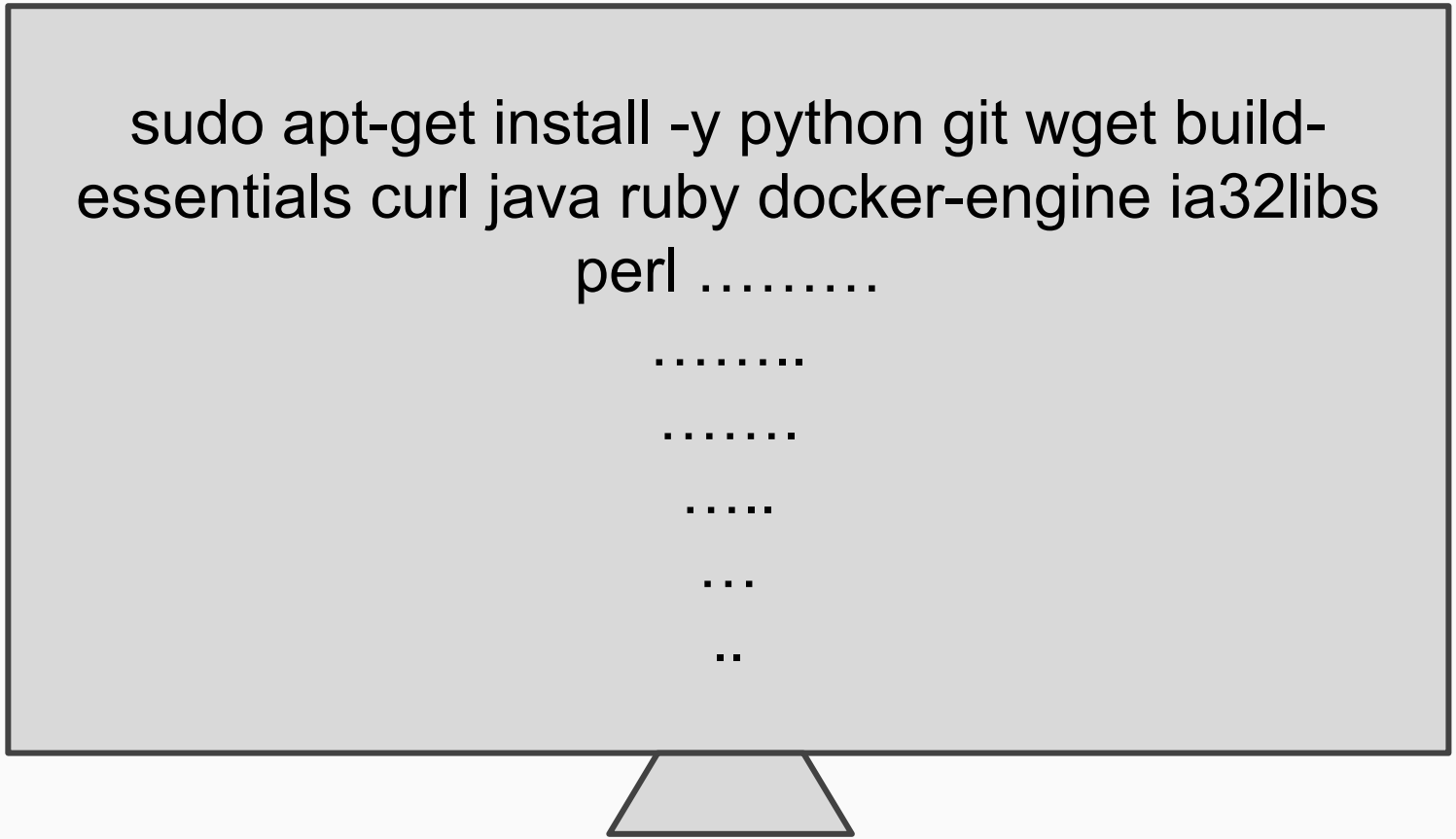
Why Microservices ?

- To deliver software faster & embrace newer technologies quickly
- To focus **Service boundaries** as Business boundaries, thereby resisting the temptation of a Service growing too large
- As a Rule of Thumb, a Microservice should be developed **within 2 weeks**
- A Microservice can be deployed on a PaaS or as part of an IaaS
- Even with a small code change, a Microservice can be **deployed independently**, without affecting other critical components in the System
- Even if one Microservice fails, the **failure doesn't cascade**, letting us isolate the problem quicker than in a Monolithic design

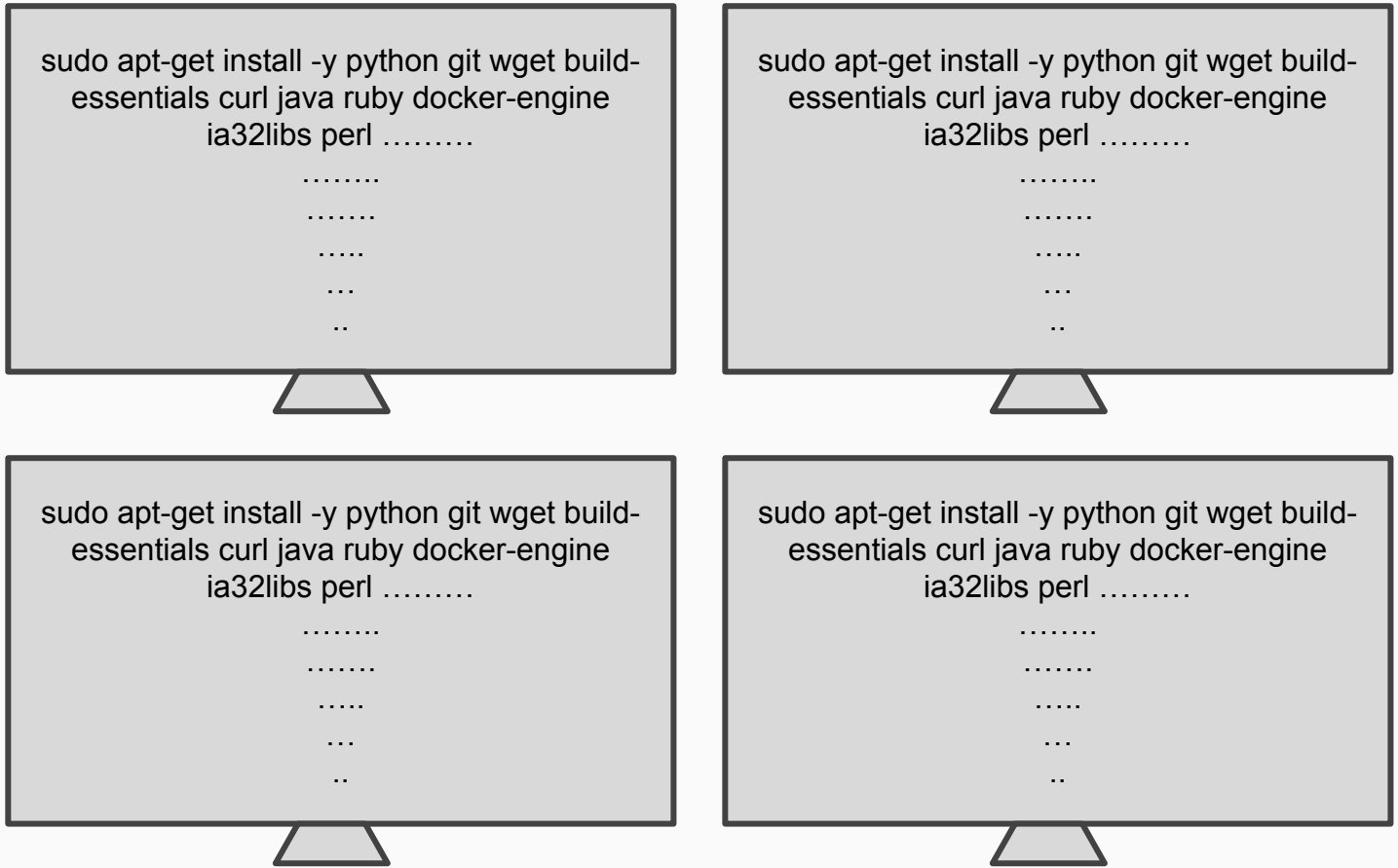
Why Microservices ? (Cont)

- Microservices can be **deployed frequently with faster bug fixing**, which isn't the case in a Monolithic design. Thus between two releases of a Microservice too many changes won't accumulate (like in Monolithic design)
- **Simple rules of ownership** with Microservices design, with distributed teams
- There can be multiple variants of Clients using the same Microservice, whether it is through Web / Desktop / Mobile etc.
- With individual Services being small in size, **replacing** them with a better implementation or even **deleting** them altogether is going to be **hassle free**
- When a codebase is just few hundred lines long, cost of replacing it is pretty small

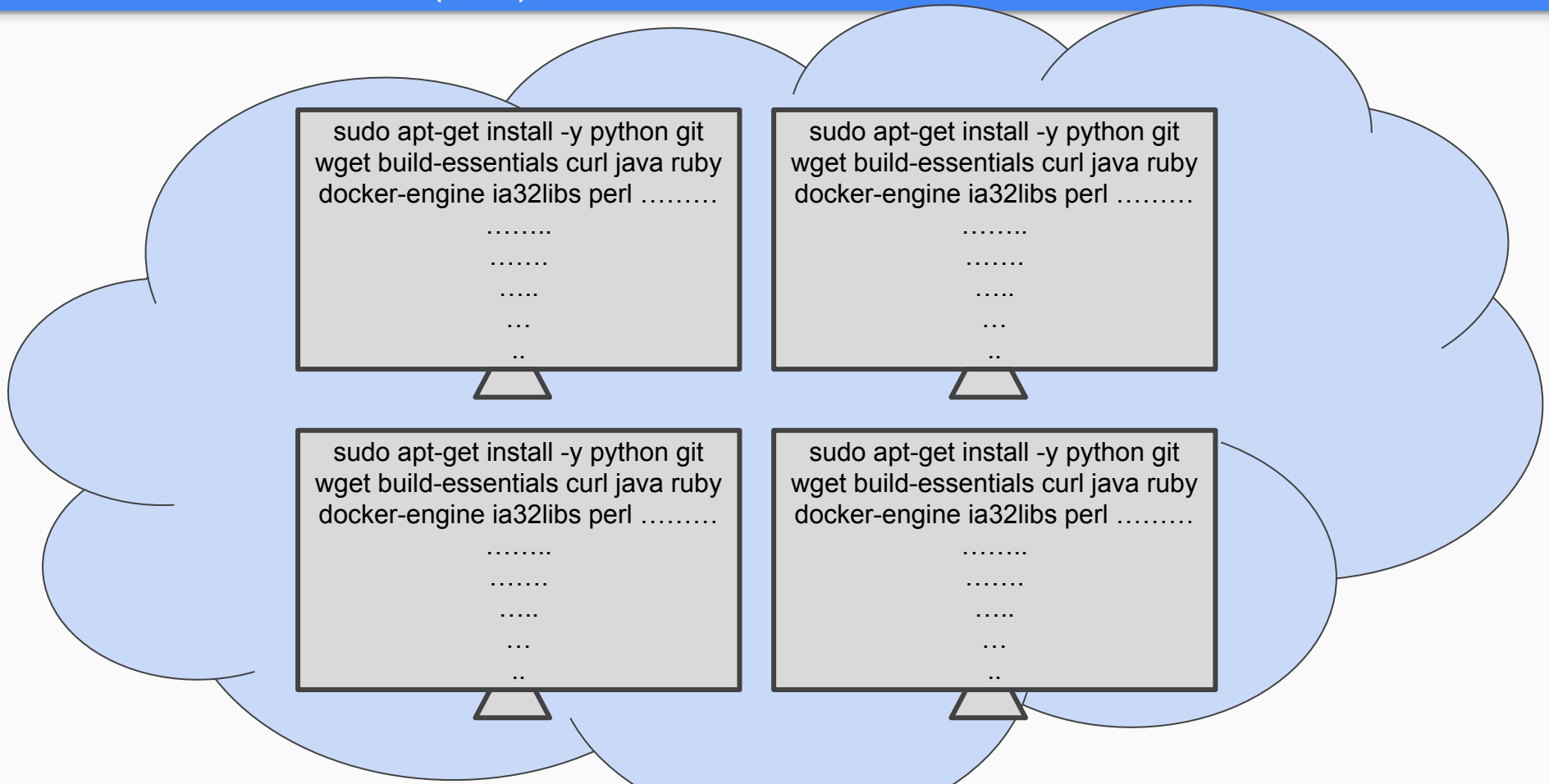
Setting up Microservices



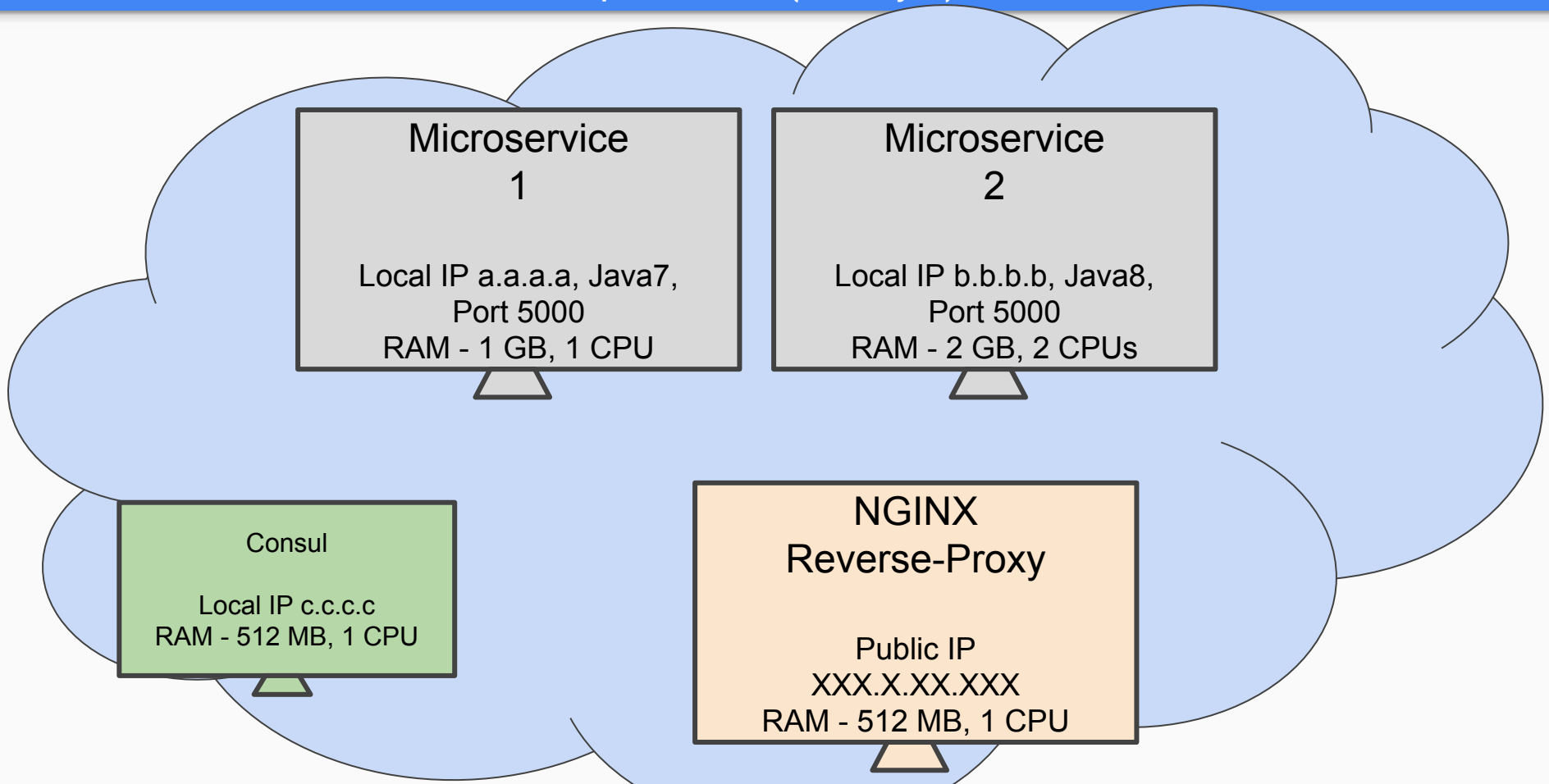
Scaling up Microservices



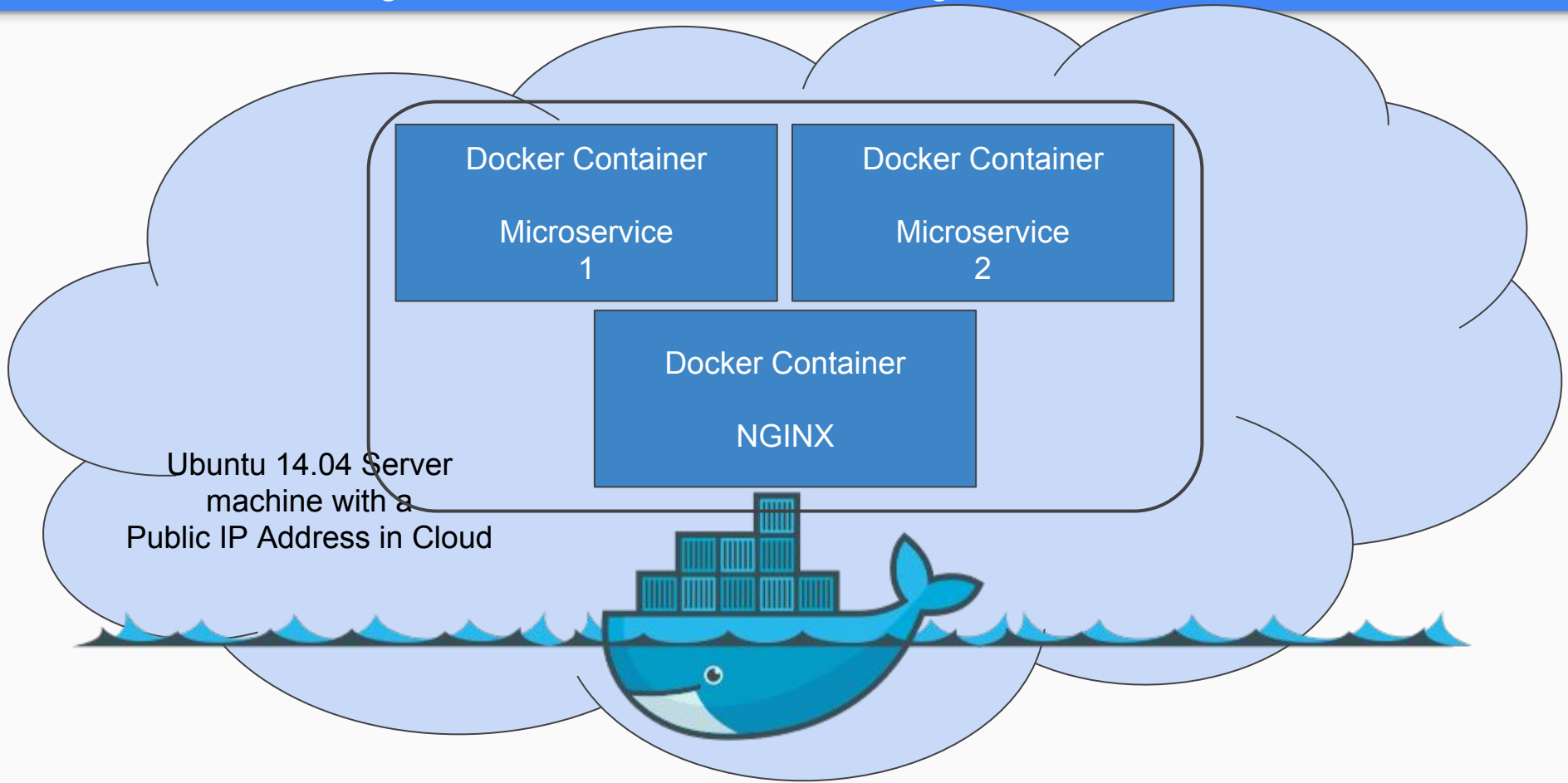
Microservices on Cloud (IaaS)



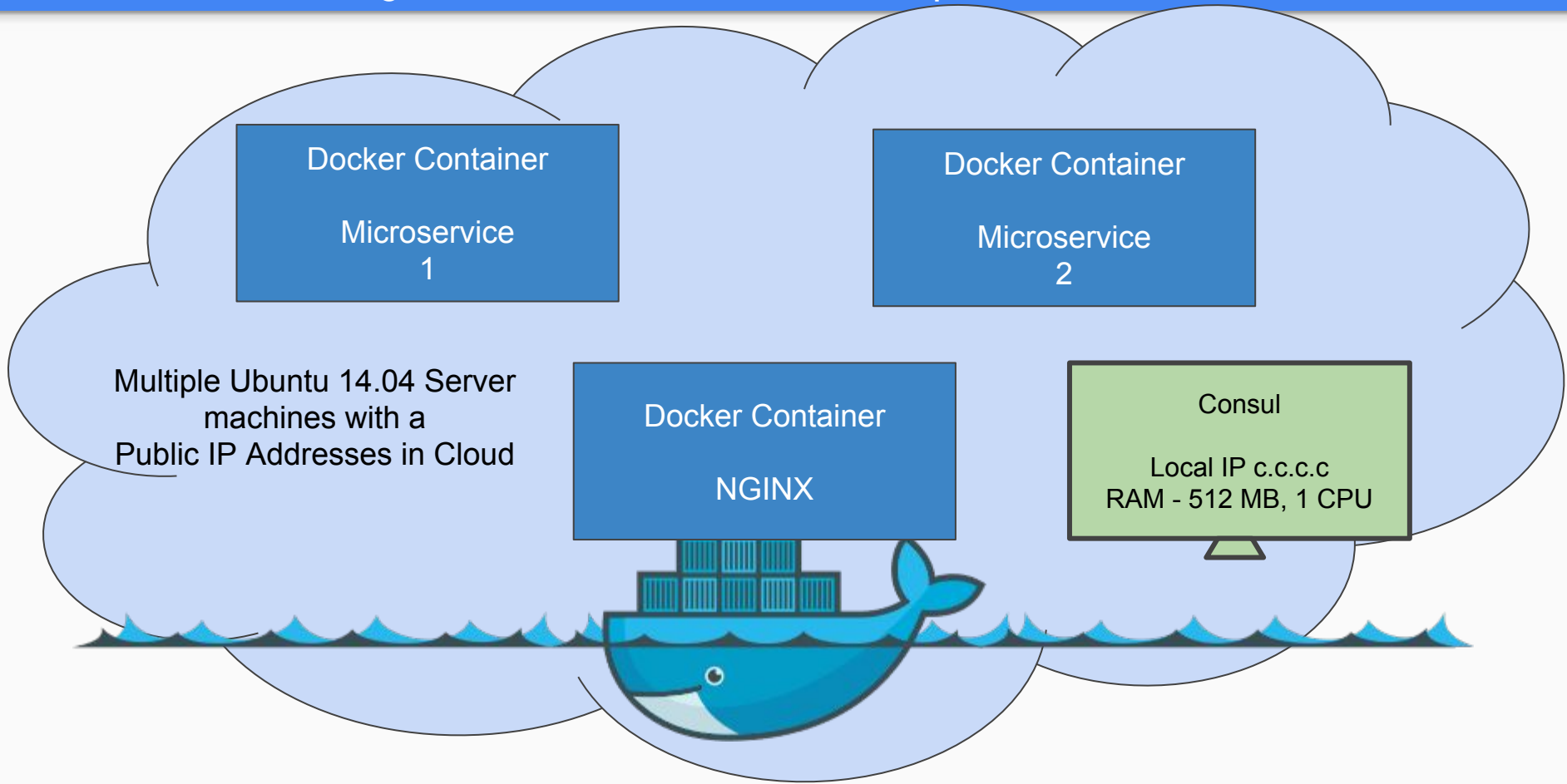
Microservices on Cloud with multiple Nodes (old style)



Microservices running in Docker Containers on a Single Ubuntu 14.04 Server Node



Microservices running in Docker Containers on multiple Ubuntu 14.04 Server Nodes



References

https://auth0.com/blog/2015/11/07/introduction-to-microservices-part-4-dependencies/

http://blog.cacoethes.co.uk/software/code-reuse-in-micro-services

http://microservices.io/patterns/microservices.html

https://www.nginx.com/blog/microservices-at-netflix-architectural-best-practices/

http://plainoldobjects.com/2015/09/02/does-each-microservice-really-need-its-own-database-2/